Text-to-Image Generation

Aleksandr Timashov
Stanford University
timashov@stanford.edu

Abstract

For the last decade we saw significant progress in Computer Vision. Image Generation is one of the direction that researchers approached. Text - guided image generation is one of the most challenging tasks in this direction.

Starting 2014, using adversarial training (GANs) was predominant approach. However for the last three years diffusion denoising approach with a score-based loss function became notable as well: many researches tackling problem of image generation report SOTA results.

1. Introduction

There has been a huge progress in Image-to-Text generation (image captioning) over the last ten years. Most of work used an idea of decoder-encoder network where decoder is represented by one of pretrained CNN backbones (f.i. Faster-RCNN) and decoder is either RNN-based network or Transformer.

Inverse problem of image generation based on the text caption is a common challenge in Computer Vision community as well. Last month OpenAI released their work ”DALL-E 2” that can create original, realistic images and art from a text description. Below is a an image generated by the description: ”An astronaut playing basketball with cats in space”.

Two weeks later, Google Research released their work ”Imagen” that outperformed ”DALL-E 2” and attained new a new state-of-the-art results in FID score. Results were measured on COCO dataset. [13] Both models demonstrates the power of Diffusion Models in Image generation.

Existing generative modeling applies two major approaches:

- **explicit** data distribution modeling (autoregressive models, normalizing flow models, variational aitoencoders, etc.)

- **implicit** data distribution modeling (generative adversarial networks)

Explicit models directly learn the distribution's probability density via maximum likelihood estimation. Typically it is done by maximizing not density itself but some lower bound instead.

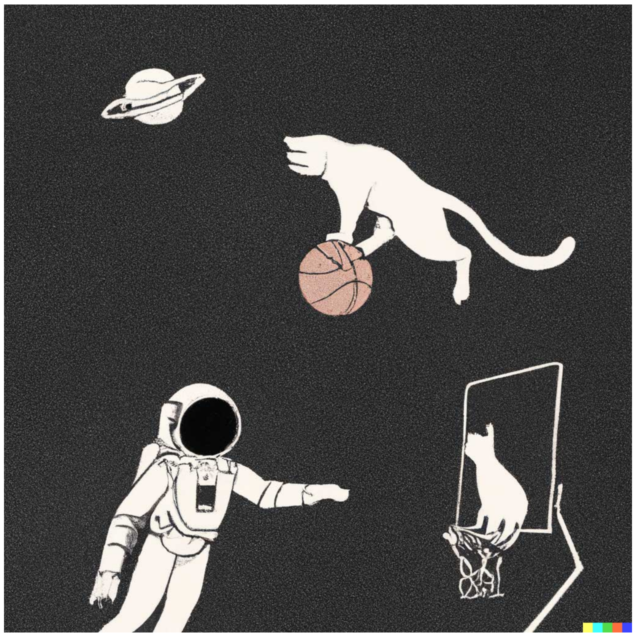


Figure 1. Image Source: https://openai.com/dall-e-2/

The key difference of diffusion models compare with other explicit models is we model not some direct output (image or text caption itself), but the gradient of the log probability density function (a score function). On the other hand, score-based approach is more stable and easier to train than GANs. Adversarial training is known for its unstable training process.

In all models that I trained for this project, input is an image and output is a score function that has the same dimension as an input. Training is done via score matching. After model is optimized, I apply sampling using Langevin dynamics. It is an iterative procedure that allows to draw samples from distribution $\mathbf{p}(\mathbf{x})$ using only its score function $\nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x})$. This procedure starts from noise.

2. Related Work

In this section I am going to discuss two types of papers:

- Diffusion models
- Text-to-Image generation

2.1. Diffusion models

Starting 2020, diffusion models with denoising (DDPM and DDIM) have seen significant success in image generation demonstrating SOTA results and often surpassing adversarial approaches (GANs) in fidelity and diversity. [5, 7, 9–11]

The paper that inspired this project the most is "Score-Based Generative Modeling through Stochastic Differential Equations" (Yang Song, at al., 2021) Later, in 2021, Song et al. [7]
In this paper, authors focuses on describing training and sampling processes:

- **training:** a forward SDE process that diffuses a data point into random noise;
- **sampling:** reverse SDE process that transforms noise back into a data point;

The reverse SDE process can be derived from the forward SDE process given the score of the marginal probability densities as a function of time. In addition to any general-purpose SDE solver to compute the reverse SDE process, the paper proposes two sampling methods, the predictor-corrector sampling and deterministic samplers based on the probability flow ordinary differential equation.
The paper also talks about controllable generation. If we have two random variables $\mathbf{x}$ and $\mathbf{y}$, and we know the forward process of generating $\mathbf{y}$ from $\mathbf{x}$, represented by conditional probability distribution $\mathbf{p}(\mathbf{y} \mid \mathbf{x})$, we can easily find $\mathbf{p}(\mathbf{x} \mid \mathbf{y})$, applying Bayes' rule:

$$\mathbf{p}(\mathbf{x} \mid \mathbf{y}) = \mathbf{p}(\mathbf{x}) \cdot \mathbf{p}(\mathbf{y} \mid \mathbf{x}) / \int \mathbf{p}(\mathbf{x})\mathbf{p}(\mathbf{y} \mid \mathbf{x})\mathbf{dx} \quad (1)$$

To simplify this expression it is proposed to take gradients with respect to $\mathbf{x}$:

$$\nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x}) + \nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{y} \mid \mathbf{x}) \quad (2)$$

Finally, it is proposed to train a model to estimate the score function of the unconditional data distribution: $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x})$ and conditional model of forward process $\mathbf{p}(\mathbf{y} \mid \mathbf{x})$.
To compute posterior function $\nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x} \mid \mathbf{y})$ is easy from the equation **(2)**.

2.2. Text-to-Image generation

Diffusion models have seen significant success in text-to-image generation as well. [1, 2, 8, 8] "DALL-E 2" [1], uses a diffusion prior on CLIP [6] text latents, to generate high resolution images it cascaded diffusion models. GLIDE [2] also uses cascaded diffusion models for text-to-image. Imagen [4] is more intuitive, since it does not need to learn a latent prior, but uses large pretrained language models.

3. Methods

In my approach I tackle the problem of conditional image generation using Bayes' rule trick (equation (2)):

$$\nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x}) + \nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{y} \mid \mathbf{x}),$$

where prior $\mathbf{s}_\theta \approx \nabla_{\mathbf{x}} \log \mathbf{p}(\mathbf{x})$ is an unconditional score based model and $\mathbf{p}(\mathbf{y} \mid \mathbf{x})$ is a image captioning model. The details of each model, training and sampling processes are below:

3.1. Score function, modeling unconditional data distribution

3.1.1 Input

Due to limited computational resources and time limitations, I resized images to 32×32 size before sending it to the model.

3.1.2 Model Architecture

The model architecture is borrowed from "Generative Modeling by Estimating Gradients of the Data Distribution" [11]. The authors name this model Noise Conditional Score Network (NCSN).
There are **4 main details** of this architecture.

- Geometric progression of added noises to perturb data
- U-Net architecture
- Instance normalization
- Dilated convolutions

One of the challenges is the estimated score functions are not accurate in low density regions. This problem exists because we have only a few data points that are available in these regions during optimization. The well known solution is to add some noise to the data points. The trade-off here is if we add more noise, we have more accurate score functions but worse quality of data samples. And from the other hand, if we add less noise, we will have less accurate score functions, but better samples in theory. The solution proposed in [11] is to start perturbing data distribution with more noise and decrease it with the following steps. More precisely it is offered to have a positive **geometric progression** sequence $\{\sigma_i\}_{i=1}^{L}$ such that

$$\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = ... = \frac{\sigma_{L-1}}{\sigma_L} > 1$$

So, σ_1 is large enough to mitigate challenges with inaccurate values in low density regions and σ_L is small enough not to affect quality of samples.

Since we have the same dimension for input and output, the natural choice for the model architecture is **U-Net** with skip connections similar to ResNet. It is known that it helps to transfer information between layer: from shallow ones to deeper ones. I incorporate noise information by perturbing it to the feature maps of the different layers. To know more specific details of the architecture, I refer to [11].

Instance normalization operates on a single data point, it was introduced in the StyleNet paper. [3] It is beneficial to tackle our challenge because score-based model can consider σ when predicting the scores. Below is the equation:

$$\mathbf{z_k} = \gamma[\mathbf{i},\mathbf{k}]\frac{\mathbf{x_k} - \mu_\mathbf{k}}{\mathbf{s_k}} + \beta[\mathbf{i},\mathbf{k}],$$

where $\mathbf{x}$ is an input with $\mathbf{C}$ feature maps, $\mu_\mathbf{k}$ and $\mu_\mathbf{k}$ are mean and standard deviation for the $k-th$ feature map of $\mathbf{x}$. $\gamma \in \mathbb{R}^{L\times C}$ and $\beta \in \mathbb{R}^{L\times C}$ are learnable parameters.
It is easy to see that it removes the information of mean for different feature maps. To tackle this issue, authors proposed to add one more term:

$$\mathbf{z_k} = \gamma[\mathbf{i},\mathbf{k}]\frac{\mathbf{x_k} - \mu_\mathbf{k}}{\mathbf{s_k}} + \beta[\mathbf{i},\mathbf{k}] + \alpha[\mathbf{i},\mathbf{k}]\frac{\mu_\mathbf{k} - \mathbf{m}}{\mathbf{v}},$$

where $\mathbf{m}$ and $\mathbf{v}$ are mean and standard deviation of μ_k respectively.

Dilated convolution is a technique of inserting spaces between kernel elements for upsampling. It is firstly applied in semantic segmentation and demonstrated grate performance. It is beneficial in our use case as well.

3.1.3 Loss function

I use a weighted sum of Fisher divergences as the most recommended objective in a score-based models:

$$\mathbf{\Sigma_{i=1}^{L}}\lambda(\mathbf{i})\mathbb{E}_{\mathbf{p}_{\sigma_\mathbf{i}}(\mathbf{x})}[||\nabla_\mathbf{x}\log\mathbf{p}_{\sigma_\mathbf{i}}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x},\mathbf{i})||_\mathbf{2}^\mathbf{2}]$$

After choosing $\lambda(i) = \sigma_i^2$ and applying integration by parts, we get the final objective:

$$\frac{\mathbf{1}}{\mathbf{L}}\mathbf{\Sigma_{i=1}^{L}}[||\sigma_\mathbf{i}\mathbf{s}_\theta(\mathbf{x_i} + \sigma_\mathbf{i}\mathbf{z_i}, \sigma_\mathbf{i}) + \mathbf{z_i}||_\mathbf{2}^\mathbf{2}]$$

3.2. Image captioning

3.2.1 Basic information

When $\mathbf{y}$ represents text captions, I can train a time-dependent image-to-text model $\mathbf{m_t}(\mathbf{x}(\mathbf{t}),\mathbf{y})$. Since the model is tractable, I can easily create a pair of training data $(\mathbf{x}(\mathbf{t}),\mathbf{y})$ by first sampling $(\mathbf{x}(\mathbf{0}),\mathbf{y})$ from COCO dataset [13] and then obtaining $(\mathbf{x}(\mathbf{t}),\mathbf{y})$ by perturbing noise.

To keep the same input size as for described above score function, I resize images to 32×32 size. For text-token embeddings, I use GloVe pretrained embeddings. For the image embeddings, I use pretrained on ImageNet ResNet model. The image captioning model architecture is similar to that one we used in **HW3** with transformers, I use 2 heads for attention. As a loss function I use temporal softmax loss. The key difference is I perturb data with a noise using Gaussian Fourier Projections.

3.2.2 Probability distribution function representation

To recall, at every timestep I produce a score for each word in the vocabulary and sample the work with the highest score. Using Bayes' rule, I can write down equation

$$\mathbf{p}(\mathbf{y_1},\mathbf{y_2},...,\mathbf{y_k}|\mathbf{x}) = \mathbf{p}(\mathbf{y_1}|\mathbf{x})\cdot ... \cdot \mathbf{p}(\mathbf{y_k}|\mathbf{y_1},\mathbf{y_2},...,\mathbf{y_{k-1}},\mathbf{x})$$

To make it simple, I can apply logarithm. To get a score function, I should find derivative with respect to $\mathbf{x}$:

$$\nabla_\mathbf{x}\log\mathbf{p}(\mathbf{y_1},\mathbf{y_2},...,\mathbf{y_k}|\mathbf{x}) = \nabla_\mathbf{x}\log\mathbf{p}(\mathbf{y_1}|\mathbf{x}) +$$

$$+\mathbf{\Sigma_{i=2}^{k}}\nabla_\mathbf{x}\log\mathbf{p}(\mathbf{y_i}|\mathbf{y_1},\mathbf{y_2},...,\mathbf{y_{i-1}},\mathbf{x})$$

Each part of this sum, we can easily find using a backpropagation.

3.3. Sampling

After score-based model s_θ is trained, I am using Annealed Langevin dynamics [11] to draw samples from this model, using the following equation:

$$\bar{\mathbf{x}}_{\mathbf{i+1}} \leftarrow \bar{\mathbf{x}}_\mathbf{i} + \frac{\alpha_\mathbf{i}}{\mathbf{2}}\nabla_\mathbf{x}\log\mathbf{p}(\mathbf{x}) + \sqrt{\alpha_\mathbf{i}}\mathbf{z_i} \quad (3)$$

Where $\nabla_\mathbf{x}\log\mathbf{p}(\mathbf{x})$ is a trained score-based model, $\mathbf{z_i}$ is an added noise which is decreasing with each step. We start sampling from complete noise $\mathbf{x}_0$. Below is the detailed algorithm.

4. Dataset and Features

For training and evaluating text2image and image2text models I should construct datasets using Vision Language data i.e., image-text pairs. Vision data is represented by raw

Algorithm 1 Annealed Langevin dynamics. ([11])

```
Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$
  Initialize $\bar{x}_0$
  for $i \leftarrow 1$ to $L$ do
    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$
    for $t \leftarrow 1$ to $T$ do
      Draw $\mathbf{z_t} \sim \mathcal{N}(0, I)$
      $\bar{x}_t \leftarrow \bar{x}_{t-1} + \frac{\alpha_i}{2} s_\theta(\bar{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\mathbf{z_t}$
    end for$\bar{\mathbf{x}}_\mathbf{0} \leftarrow \bar{\mathbf{x}}_\mathbf{T}$
  end for
  return $\bar{\mathbf{x}}_\mathbf{T}$
```

image data (or links to download this data), and Language data is represented by captions i.e., plain texts. For the sake of reproducibility, I used COCO Captions Dataset [13] contains over $330,000$ images and more than 1.5 million captions describing these images. The split for train / validate test is default as follows:

- **train:** $\approx 120K$ images
- **validate:** $\approx 5K$ images
- **test:** $\approx 205K$ images

For the follow up work it might make sense to make a different split and use Conceptual Captions dataset [12] form Google AI contains over 3 million images (i.e. links to download them), paired with language captions (represented by TSV file).

Due to the challenges with computations I resized images to 32×32 size. Below are the examples of images from the dataset:

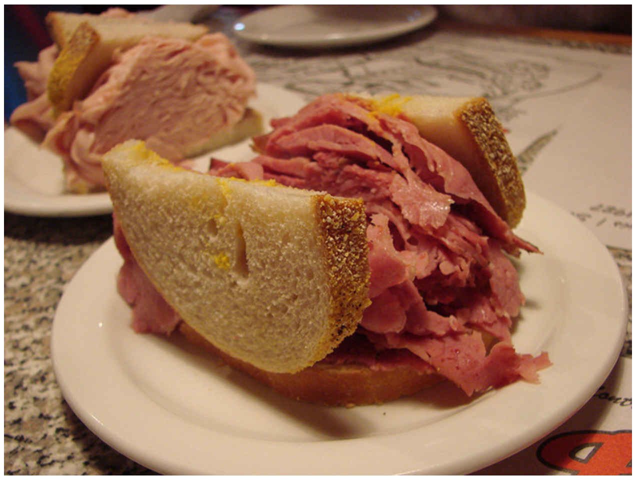
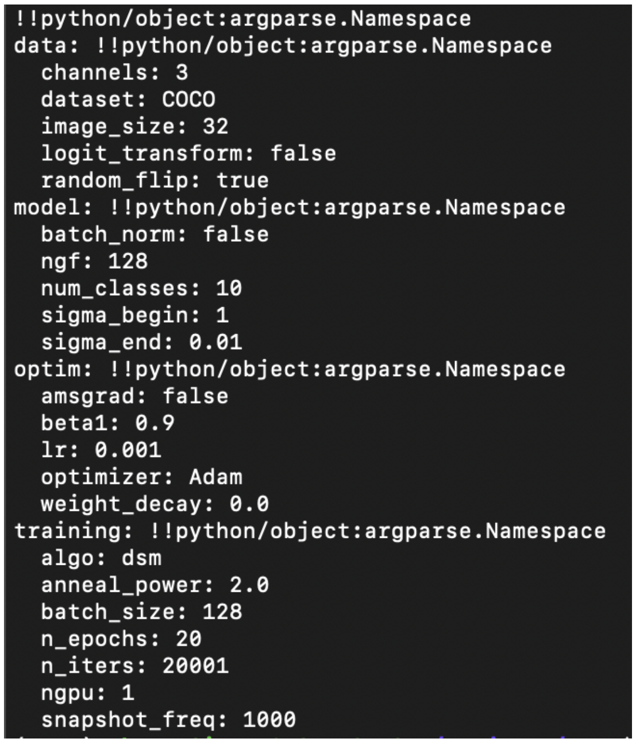


Figure 2. Image Source: COCO dataset

5. Experiments/Results

5.0.1 Configuration

Below are the configurations for my the most successful experiment. I trained model in batches with 128 images in



```
!!python/object:argparse.Namespace
data: !!python/object:argparse.Namespace
  channels: 3
  dataset: COCO
  image_size: 32
  logit_transform: false
  random_flip: true
model: !!python/object:argparse.Namespace
  batch_norm: false
  ngf: 128
  num_classes: 10
  sigma_begin: 1
  sigma_end: 0.01
optim: !!python/object:argparse.Namespace
  amsgrad: false
  beta1: 0.9
  lr: 0.001
  optimizer: Adam
  weight_decay: 0.0
training: !!python/object:argparse.Namespace
  algo: dsm
  anneal_power: 2.0
  batch_size: 128
  n_epochs: 20
  n_iters: 20001
  ngpu: 1
  snapshot_freq: 1000
```

Figure 3. Run configuration.

one batch. I targeted to use 20 epochs, but after 12 of them it was clear that model does not converge anymore. As an optimizer I used Adam with learning rate $1e-3$.

5.0.2 Loss Convergence

Loss function for score matching is represented below.

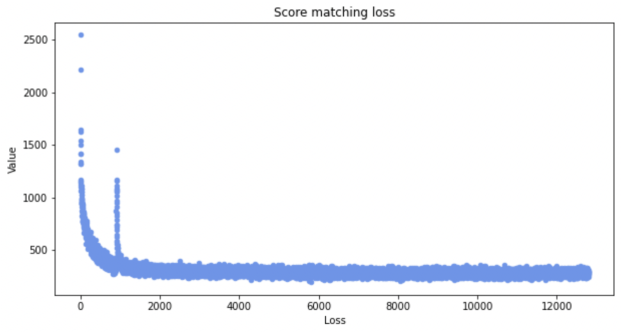


Figure 4. Loss function for score matching

As we can see it is decreasing. Around step number $1,000$

we can see that our optimizer lost local minimum, but after that started converging again.

5.0.3 Results

Below are some results starting from noise and converging to clear pictures.

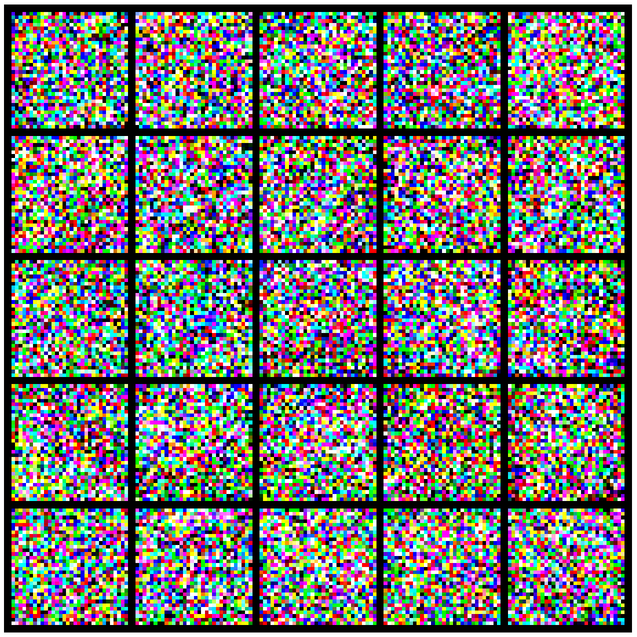


Figure 5. Step 18.

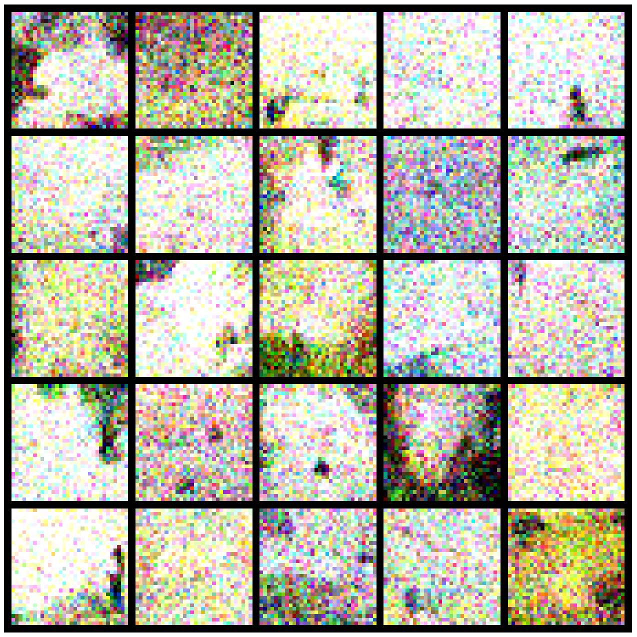


Figure 6. Step 348.

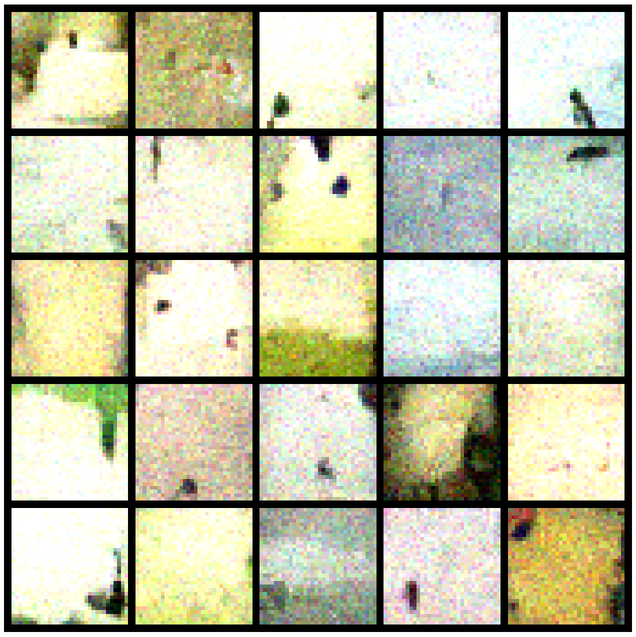


Figure 7. Step 548.

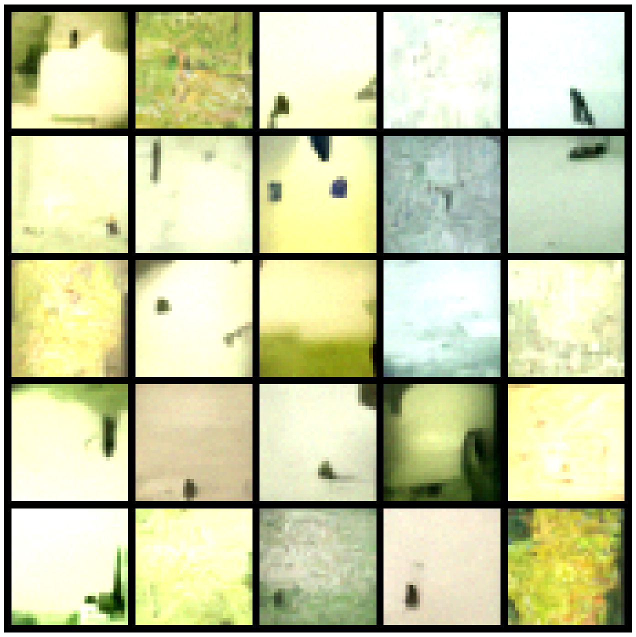


Figure 8. Step 999.

6. Conclusion/Future Work

My work shows the great opportunity to experiment with image generation using score-based models with denoising score matching. I experimented using different model architectures, including DDPM and NCSN. Sampling was done using Annealed Langevin dynamics.

Current experiments showed promising direction. However there is a huge room for improvements:

- Controllable sampling from sum of prior distribution and trained caption model
- Experiments with increased input size
- Experiments with different models architectures
- Experiments with Conceptual Captions dataset [12]

References

[1] Alex Nichol Casey Chu Mark Chen Aditya Ramesh, Prafulla Dhariwal. Hierarchical text-conditional image generation with clip latents, 2022. 2

[2] Aditya Ramesh Pranav Shyam Bob McGrew Pamela Mishkin Ilya Sutskever Alex Nichol, Prafulla Dhariwal and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2021. 2

[3] Min Sun† Ming-Yu Liu Cheng-Kuan Chen, Zhu Feng Pan. Unsupervised stylish image description generation via domain layer norm, 2018. 3

[4] William Chan et al. Chitwan Saharia. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 2

[5] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2022. 2

[6] A Radford et al. Learning transferable visual models from natural language supervision, 2021. 2

[7] Yang Song et al. Score-based generative modeling through stochastic differential equations, 2021. 2

[8] Taesung Kwon et al. Gwanghyun Kim. Diffusionclip: Text-guided diffusion models for robust image manipulation, 2021. 2

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2

[10] William Chan David J Fleet-Mohammad Norouzi Jonathan Ho, Chitwan Saharia and Tim Salimans. Cascaded diffusion models for high fidelity image generation, 2022. 2

[11] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019. 2, 3, 4

[12] Nan Ding Radu Soricut Soravit Changpinyo, Piyush Sharma. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts, 2021. 4, 6

[13] Serge Belongie Lubomir Bourdev-Ross Girshick James Hays Pietro Perona Deva Ramanan C. Lawrence Zitnick Piotr Dollár Tsung-Yi Lin, Michael Maire. Microsoft coco: Common objects in context, 2015. 1, 3, 4