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 autoencoders, 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.


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 $p(x)$ using only its score function $\nabla_x \log p(x)$. 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] proposed in [11] is to start perturbing data distribution with score functions, but better samples in theory. The solution 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 gradients demonstrating SOTA results and often surpassing adversarial approaches (GANs) in fidelity and diversity.

\[ \nabla \log p(x) = \nabla \log p(x) + \nabla \log p(y | x) \]  (2)

Finally, it is proposed to train a model to estimate the score function of the unconditional data distribution: \( s_{\theta}(x) \approx \nabla_x \log p(x) \) and conditional model of forward process \( p(y | x) \).

To compute posterior function \( \nabla_x \log p(x | y) \) 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)):

\[ \nabla \log p(x | y) = \nabla \log p(x) + \nabla \log p(y | x) \]

where prior \( s_{\theta} \approx \nabla_x \log p(x) \) is an unconditional score based model and \( p(y | x) \) 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 \( \{\sigma_l\}_{l=1} \) such that

\[
\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = \ldots = \frac{\sigma_{L-1}}{\sigma_L} > 1
\]

So, \( \sigma_1 \) is large enough to mitigate challenges with inaccurate values in low density regions and \( \sigma_L \) 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 \( \sigma \) when predicting the scores. Below is the equation:

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

where \( x \) is an input with \( C \) feature maps, \( \mu_k \) and \( \mu_k \) are mean and standard deviation for the \( k \)-th feature map of \( x \). \( \gamma \in \mathbb{R}^{L \times C} \) 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:

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

where \( m \) and \( v \) are mean and standard deviation of \( \mu_k \) 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:

\[
\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(x)}[||\nabla_x \log p_{\sigma_i}(x) - s_\theta(x, i)||^2]
\]

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

\[
\frac{1}{L} \sum_{i=1}^{L} [||\sigma_l s_\theta(x_i + \sigma_i z_i, \sigma_i) + z_i||^2]
\]

3.2. Image captioning

3.2.1 Basic information

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

To keep the same input size as for described above score function, I resize images to \( 32 \times 32 \) 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

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

To make it simple, I can apply logarithm. To get a score function, I should find derivative with respect to \( x \):

\[
\nabla_x \log p(y_1, y_2, ..., y_k|x) = \nabla_x \log p(y_1|x) + \sum_{i=2}^{k} \nabla_x \log p(y_i|y_1, y_2, ..., y_{i-1}, x)
\]

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

3.3. Sampling

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

\[
\bar{x}_{i+1} \leftarrow \bar{x}_i + \frac{\alpha_i}{2} \nabla_x \log p(x) + \sqrt{\sigma_i} z_i
\]

Where \( \nabla_x \log p(x) \) is a trained score-based model, \( z_i \) is an added noise which is decreasing with each step. We start sampling from complete noise \( x_0 \). 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

\( \alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2 \)

end for

for \( t \leftarrow 1 \) to \( T \) do

Draw \( z_t \sim N(0, I) \)

\( \bar{x}_t \leftarrow \bar{x}_{t-1} + \frac{\alpha_t}{2} s_\theta(\bar{x}_{t-1}, \sigma_t) + \sqrt{\alpha_t} z_t \)

end for

end for

\( \bar{x}_0 \leftarrow \bar{x}_T \)

return \( \bar{x}_T \)

For the follow up work it might make sense to make a different split and use Conceptual Captions dataset [12] from 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 \( 32 \times 32 \) size. Below are the examples of images from the dataset:

![Figure 2. Image Source: COCO dataset](image)

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 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 \( 1 \epsilon - 3 \).

5.0.2 Loss Convergence

Loss function for score matching is represented below.

![Figure 3. Run configuration.](image)

![Figure 4. Loss function for score matching.](image)

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

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