IMAGE GENERATION USING TRANSFORMERS

Lyuheng Yuan
lyuheng@seas.upenn.edu

December 16, 2020

ABSTRACT

We compare Transformer and typical autoregressive convolutional models in image generation to see whether self-attention architecture can outperform previous methods qualitatively and quantitatively. By restricting receptive field to local neighbourhoods, our model can achieve comparable performance but ends up using significantly less parameters and model size on CIFAR-10. We also test model capability by image completion and derive reasonable generated image conditioned on top-half part. Code is available at https://github.com/lyuheng/546project.

***K*eywords** Autoregressive model · PixelCNN · Transformer

1 Introduction

Generative models have been widely used in image denoising, deblurring, inpainting that rely on generating images conditioned on noisy or incomplete data. The nice self-attention property can make use of long-term information to capture long-range common phenomena in images, such as symmetry and occlusion. This architecture could be viewed as a strong variant of RNN but highly paralleled.

Some models, such as PixelCNN++[2], have a tractable log-likelihood which reflects the the capacity in modeling the distribution of natural images. One possible disadvantage of CNN based models lies in the limited size of receptive field that can adversely affects their ability in recognizing long-term pattern. If we stack more layers to grow receptive field, we end up significantly increasing the parameters and model size. Meanwhile, this would be also computationally costly in training such models.

There have been some works in modeling RGB image in raster-scan order as sequential signal and use RNNs for modeling the conditional distribution. Although temporal architectures have unlimited receptive field, they are highly unparalleled since the latter token is computed based on former results and it would take long time to train. Self-attention can achieve a better balance in the trade-off between the virtually unlimited receptive field of the necessarily sequential RNNs and the limited receptive field of the much more paralleled PixelCNN and its various extensions.

Transformer[3, 4] entirely uses self-attention layer together with layer normalization and dropout, and shows great capacity in language analysis and understanding. The way how it accelerates standard RNNs is through paralleled matrix multiplication and shared feed-forward networks. However, we are still unsure about how well temporal models are generalized into image domain since it is not completely convincing to treat a RGB image in raster-scan order.

2 Background & Related Work

Convolutional Model The autoregressive model proposed in PixelCNN[1] by using Gated Convolutional Layer enables us to model more complex interactions, similar to gates in LSTM or GRU conceptually. Its variant PixelCNN++[2] chooses a continuous logistic distribution for us to simply compute a smooth and memory efficient pixel distributions, instead of a 256-way softmax. This is advantageous in reducing memory cost and reasonable in classifying pixel distribution. Its shortcut connections can recover information which might be lost during subsampling and upsampling. PixelCNN++ uses over 30 convolutional layers in total and achieve a negative log-likelihood of 2.94 on CIFAR-10. We use this model as a baseline in the following experiments.

Generative Adversarial Network Another popular image generation technique is training models through adversarial loss[5]. In other words, generator is trained to output images that can fool the discriminator that is designed to determine whether a given image in real or fake. While GANs can generate quite realistic images, they also have some drawbacks. GANs are difficult to train and suffer from mode collapsing. Since GANs are implicit generative models that do not have tractable log-likelihood, it is often difficult to tune hyperparameters in terms of comparing performance.

3 Approach

3.1 Discretized Logistic Mixture Likelihood

For a certain sub-pixel, we consider using a logistic distribution ν to represent its density parameterized by μ, s, i.e.

$$\nu \sim \textit{logistic}(\mu, s) \tag{1}$$

Since each sub-pixel follows a discrete distribution over $[0, 255]$, we round this value which forms a 'discretized logistic distribution'(CDF):

$$P(x|\mu, s) = \begin{cases} \sigma(\frac{x-\mu+0.5}{s}) & \text{if } x = 0 \\ \sigma(\frac{x-\mu+0.5}{s}) - \sigma(\frac{x-\mu-0.5}{s}) & \text{if } 0 < x < 255, x \in \mathbf{N} \\ 1 - \sigma(\frac{x-\mu-0.5}{s}) & \text{if } x = 255 \end{cases} \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function. We can extend the assumption Eqn.(1) to multi-modal by using a mixture of logistics for ν, i.e.

$$\nu \sim \sum_{i=1}^{K} \pi_i \textit{logistic}(\mu_i, s_i) \tag{3}$$

where weights π_i sums up to 1.

Another intuitive way is to classify value of sub-pixel by a 256-way softmax. Although this gives the model flexibility, it is also costly in memory. In total, there are $256 \times 3 = 768$ parameters for each pixel in categorical distribution, whereas we only output 100 parameters for $K = 10$ mixture components. So it's a $7\times$ reduction in memory. Moreover, the model does not know that a value of 128 is close to a value of 127 or 129, and it would assign a low probability to pixels observed in low frequencies, which is problematic. So we adopt discretized logistic mixture in our experiment.

3.2 Loss Function

Generally speaking, we would like to maximize the log-likelihood $\log p(x) = \sum_{t=1}^{h \cdot w \cdot 3} \log p(x_t | x_{<t})$ for every image. The generation of each pixel depends on its formal pixels and we replicate this process until we derive a complete image. Since every sub-pixel x is scaled to $x' \in [-1, 1]$ in our implementation, we consider four cases in this range.

- Black Pixel ($x' < -0.999$, $x = 0$) $l_b = \log(\sigma(\frac{x-\mu+0.5}{s}))$.
- White Pixel ($x' > 0.999$, $x = 255$) $l_w = \log(1 - \sigma(\frac{x-\mu-0.5}{s}))$.
- Overflow Condition ($\sigma(\frac{x-\mu+0.5}{s}) - \sigma(\frac{x-\mu-0.5}{s}) < 10^{-5}$) This case happens when the centre of logistic distribution is far away from the pixel range $[-1, 1]$. Therefore the difference of two sides would become 0 and if we take the logarithm, we get $-\infty$. To solve this problem, we can instead approximate the integral by taking the PDF of the logistic and multiply it by a pixel width interval, i.e., $l_o = \log(pdf \cdot \frac{1}{127.5}) = \log(\frac{\exp(-(x-m)/s)}{s(1+\exp(-(x-m)/s))^2}) - \log(127.5)$.
- Standard Scenario (otherwise) $l_s = \log(max(\sigma(\frac{x-\mu+0.5}{s}) - \sigma(\frac{x-\mu-0.5}{s}), e^{-12}))$.

Every sub-pixel should fall into one of above cases. Given its two parameters μ and s, we are able to compute its log-likelihood. Therefore log-likelihood of entire image could be written as

$$\log p(x) = \log \sum_{i=1}^{h \cdot w} \exp\{\sum_{j=1}^{C=3} (c_{j,b}^i l_b^i + c_{j,w}^i l_w^i + c_{j,o}^i l_o^i + c_{j,s}^i l_s^i) + \log softmax(\pi_i)\} \tag{4}$$

where $c_{j,b}$ is a conditional factor refers to j^{th} channel and black pixel case. Notice that each sub-pixel only has 1 conditional factor equals to 1. We use bits per dimension (bits/dim) as metric to compare performance of different models.

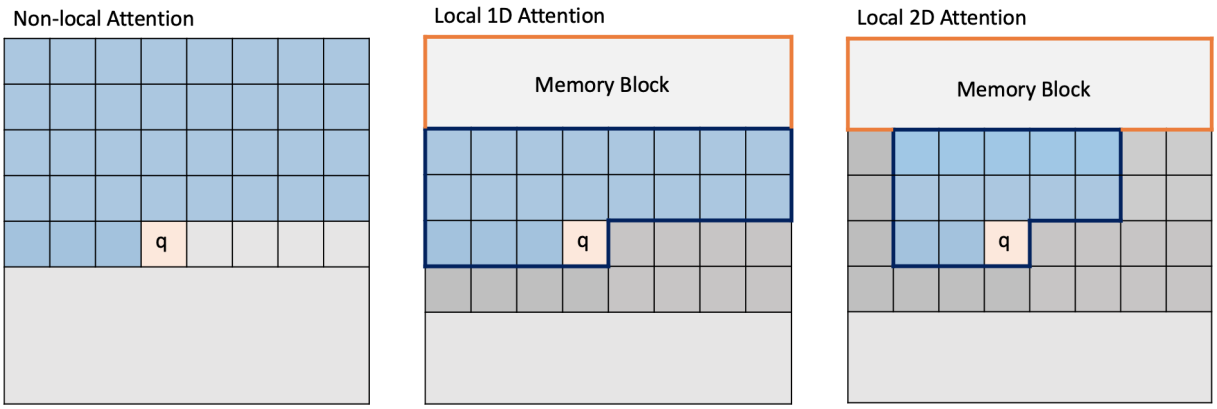

Figure 1: Different schemes of self-attention in a linearized image. The pixel marked as q is the query position. Blue pixels are where it attends to for every case. Gray pixels are masked and do not contribute to the query position. **Left**: For Non-local Attention case, q leverages all former generated sub-pixels. **Mid**: For Local 1D attention case, q only leverages pixels within current block and above neighbour block. **Right**: Inspired by CNNs, we consider using more balanced number of pixels next to and above the query position based on 1D. Kernel size here refers to how far the pixels are away from the query position we are considering.

3.3 Self-Attention

Self-attention layer computes a d-dimensional vector for each position. It first compares the position's current representation to other position's representation through 'key' and 'query', obtaining a discrete distribution over other positions. This distribution is then used to weight the contribution of the other positions'representations to the next representation for the position at hand through 'value'. The derivation of 'query','key' and 'value' are parameterized by W_q, W_k and W_v respectively. Feed-forward layers are shared across all positions parameterized by W_1 and W_2 which include short-cut connection as well. To enhance the capacity, we adopt multi-head self-attention which makes model focus on different positions at same time.

$$q_{att} = Layernorm(q + dropout(softmax(\frac{W_q q(MW_k)^T}{\sqrt{d}})MW_v)) \tag{5}$$

$$q' = Layernorm(q_{att} + dropout(W_1 relu(W_2 q_{att}))) \tag{6}$$

If we want to generate the last sub-pixel, it should attend to $32 \times 32 = 1024$ positions, which is computationally expensive. To alleviate this effect, we adopt the notion of locality. We first partition the linearized image into several non-overlapping blocks with fixed length B, padding 0 if necessary. For query of a given position, it should attend to all positions within the former block neighbor(without masks, because they have already been generated), as well as positions within current block with mask. Hence query complexity of all positions is in $\mathcal{O}(h \cdot w \cdot B \cdot d)$.

In Figure 1, we present different schemes of non-local and local self-attention. 1D local is similar as the aforementioned 'blockwise' self-attention. 2D local is based on this concept and only uses the real neighbourhoods within former and current blocks, while rest are masked. If query position moves 1 pixel to the right, all its neighbourhoods moves the same way which resembles the idea in CNN kernel.

3.4 Inference

We are able to visualize the quality of models in two ways. Firstly we can randomly sample a pixel and let the model synthesize entire image according to that value. Another way is known as image completion where half of image is given to the model and let it recover the rest part. This is a lot more meaningful because results are unsurprisingly vacuous and blurry in the first way.

Model Type	Params	Loss Function	Bits/dim ↓
PixelCNN++	–	dmol	3.09
1D local Transformer(6-layer)	block_length=256, stride = 1	dmol	3.16
2D local Transformer(6-layer)	kernel_size = 4	dmol	3.28
2D local Transformer(6-layer)	kernel_size = 6	dmol	3.23

Table 1: Bits/dim on CIFAR-10 test set. The model size of 6-layer Transformer(12MB) is much less than PixelCNN++(205MB), but ends up relatively matching its performance quantitatively.

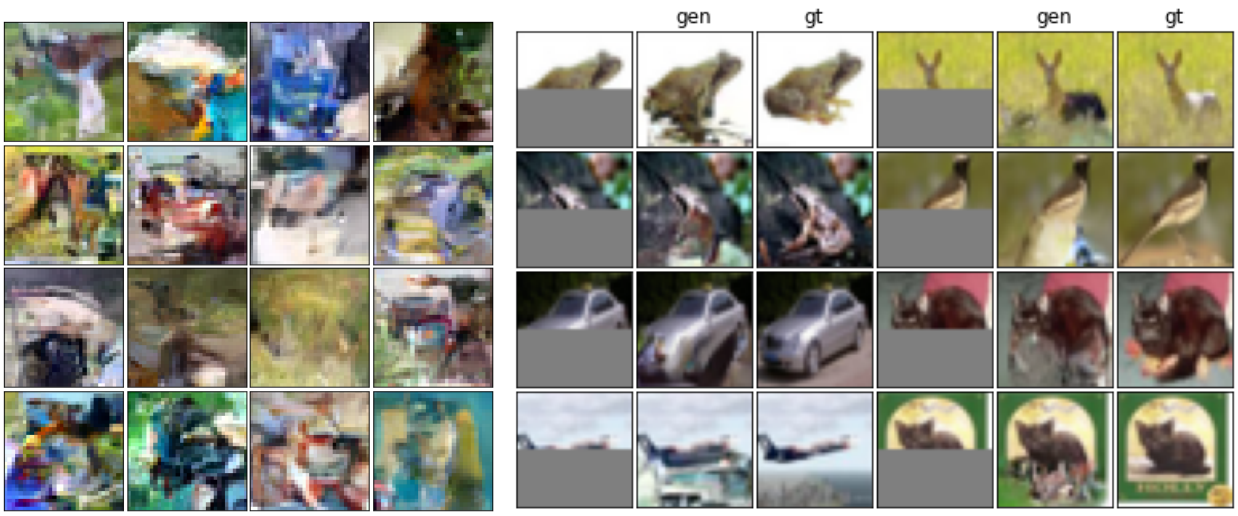

Figure 2: Generated images through PixelCNN++. **Left**: Given a random pixel to start off, we sample every sub-pixel based on this random value and prior generated values iteratively. **Right**: Given half of original image, we sample rest sub-pixels based on the the top part and prior generated values. 'gen' and 'gt' refer to generated and ground-truth respectively.

4 Experimental Results

We conduct experiments for PixelCNN++[1] and Transformer settings, both of which are trained and evaluated on CIFAR-10 dataset. Quantitative results are shown in Table 1. Our best unconditional model achieves 3.16 bits/dim on the test set using discretized mixture of logistics(dmol). We use 6 layers with $d = 256$, heads$= 8$, feed-forward dimension 512 and a dropout rate of 0.1. Block size(length) is a trade-off between linearized receptive field and memory cost. Growing of receptive field improves performance significantly. Our model uses a block length of 256.

We can view the qualitative results of both models in Figure 2, 3 and 4. While looking closer at the left generated images, they are not meaningful and realistic. Unlike GANs, sampling sub-pixels over entire image is unstable and heavily depends on prior predicted incorrect values, which resembles training RNNs without 'teacher forcing'. In both cases, synthesized images are blurry and contain various colors since we are sampling with too much randomness. For the image completion task, both models have comparable results because they have prior information on what the image looks like.

Table 1 shows that large kernel size enhances model's capacity in referring to larger receptive field, so we can see better result if we use larger kernel size. Moreover, the number of parameters used by local self-attention is independent of the size of the receptive field, which is advantageous to CNN kernel. But it might take longer time to converge.

5 Discussion

In this project, we compare the performance of convolutional based and Transformer architecture in modelling natural image through multi-modal logistic mixture. With fewer layers and less parameters, we observe comparable image

[1]We refer to OpenAI's pixelcnn++ implementation: https://github.com/openai/pixel-cnn

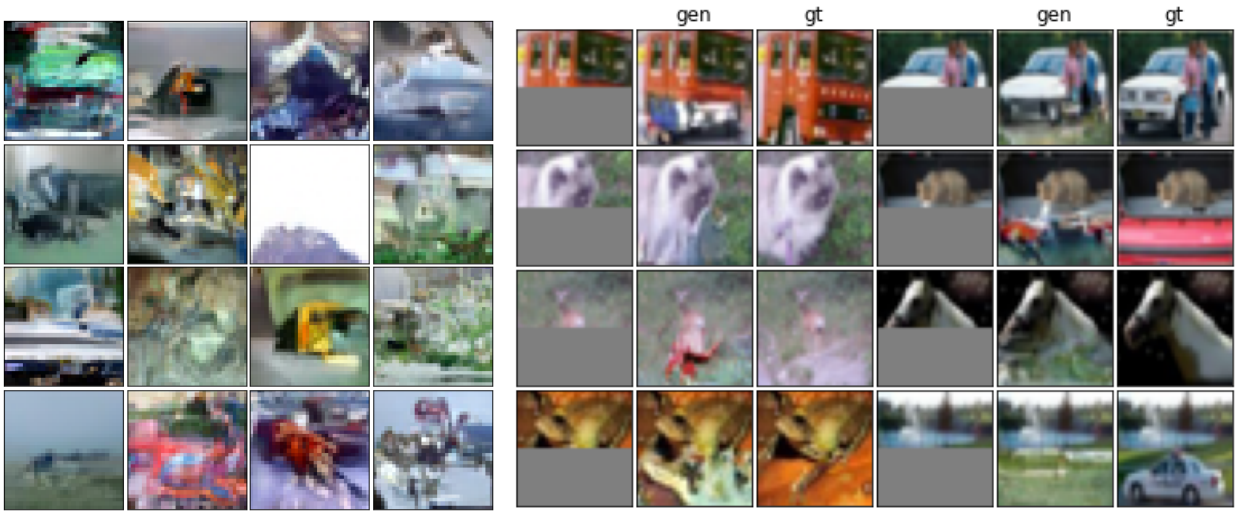

Figure 3: Generated images through 1D Transformer. **Left**: Given a random pixel to start off, we sample every sub-pixel based on this random value and prior generated values iteratively. **Right**: Given half of original image, we sample rest sub-pixels based on the the top part and prior generated values. 'gen' and 'gt' refer to generated and ground-truth respectively.

Figure 4: Generated images through 2D Transformer. **Left**: Given a random pixel to start off, we sample every sub-pixel based on this random value and prior generated values iteratively. **Right**: Given half of original image, we sample rest sub-pixels based on the the top part and prior generated values. 'gen' and 'gt' refer to generated and ground-truth respectively.

generation quality and quantitative results by negative log-likelihood. We also compare contribution of different mechanisms and hyperparameters of local self-attention that would affect final results.

In future work we would like to explore more into super-resolution which can be done by both CNN based and sequential models. We only try unconditional generation in our work, so conditional generation such as text label or images might be worthwhile to try.

References

[1] Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A. (2016). Conditional image generation with pixelcnn decoders. Advances in neural information processing systems, 29, 4790-4798.

[2] Salimans, T., Karpathy, A., Chen, X., Kingma, D. P. (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. arXiv preprint arXiv:1701.05517.

[3] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., Tran, D. (2018). Image transformer. arXiv preprint arXiv:1802.05751.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008.

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).