Taming Transformers for High-Resolution Image Synthesis

1. Intro: Attempts at Transformers in Vision

(a) Generative Pretraining from Pixels (Sutskever): Pretrain transformer by autoregressive and BERT objectives. 'GPT-2 scale' trained on low-res ImageNet gets 96p CIFAR by linear probe and 99p with fine-tuning. Adding web images gets 72p by linear probe on ImageNet.

This paper has a nice introduction about the history of pretraining strategies going out of style in generative modeling of images.

Idea is to show that in the low-resolution setting with lots of compute, generative pre-training is competitive with other self-supervised approaches.

(b) Hierarchical Autoregressive Image Models with Auxiliary Decoders: Why does likelihood loss reward local correlations > long-range structure? Review lit of models that learn local features well then train autoregressive model on top.

Extend VQ layers to other differentiable models than VAE.

AE problem I hadn't thought about: Latents contain noise since it can't be predicted from preceding pixels so can't be learned by autoregressive decoder.

2. Intro: Transformers

Each transformer layer consists of an attention mechanism followed by fully connected layer applied to all positions independently. The self attention network has the form,

$$Attn(Q, K, V) = softmax\Big(\frac{QK^t}{\sqrt{d_k}}\Big)V \in \mathbb{R}^{N\times d_v}$$

where $Q \in \mathbb{R}^{N\times d_k}$, $K \in \mathbb{R}^{N\times d_k}$ and $V \in \mathbb{R}^{N\times d_v}$

This scales like like n^2 because of the inner product. But this is worse for images because resolution introduces another n^2 scaling. You can restrict receptive field of attention modules which reduces expressiveness. Retaining full receptive field can be improved to $n\sqrt{n}$ and in practice >64 pixels is prohibitive.

3. Intro: Inductive bias and convolutions

Convolutions work by restricting to a local neighborhood, creating linear scaling with sequence length and quadratic scaling in kernel size. So CNN architectures have <3x3 kernels. This is efficient but long-range correlations are important. The concept of this paper is to combine the linear scaling of CNNs with long-range modeling of transformers.

Similar ideas are "two stage approaches". (They claim, though I'm not sure I agree)

(a) (vae on vae) A Disentangling Invertible Interpretation Network for Explaining Latent Representations Esser, Rombach, Ommer

(b) (flow on vae) Generative Latent Flow: A framework for non-adversarial image generation Xiao, Kreis, Kautz, Vahdat vanishing noise limit of vae with flow prior

4. Model: Learn a Codebook for Transformers

We need our image to be a sequence.

Represent image as a collection of codebook entries $z_q \in \mathbb{R}^{h\times w\times n_z}$ where n_z is dim of codes.

Learn a convolutional encoder E / decoder G model like in VQVAE with element-wise quantization of spatial codes $z_{ij} \in \mathbb{R}^{n_z}$ onto closest codebook entry z_k,

$$z_q = q(\hat{z}) := \Big(argmin_{z_k\in Z}||\hat{z}_{ij} - z_k||\Big) \in \mathbb{R}^{h\times w\times n_z}$$

Altogether the model looks like,

$$\hat{x} = G(q(E(x)))$$

Backprop through non-differentiable quantization operation by straight-through estimator (LOL). Loss is VQVAE loss,

$$L_{VQ}(E, G, Z) = ||x - \hat{x}||^2 + ||sg[E(x)] - z_q||_2^2 + \beta||sg[z_q] - E(x)||_2^2$$

where $sg[*]$ is stop gradient.

5. Model: Improve Codebook with GANs

We need extremely good features to model with the transformer. Enter VQGAN, a VQVAE variant that adds a discriminator and perceptual loss. Previous papers learn codebook with only a shallow model.

$$L_{GAN}(E, G, Z, D) = [\log D(x) + \log(1 - D(\hat{x}))]$$

Altogether our objective to learn an optimal compressed model Q^* is

$$Q^* = argmin_{E,G,Z} max_D \mathbb{E}_{x\ p(x)} \Big[L_{VQ}(E, G, Z) + \lambda L_{GAN}(E, G, Z, D)\Big]$$

where $\lambda = \frac{\nabla_{G_L}[L_{rec}]}{\nabla_{G_L}[L_{GAN} + \delta}$.

Apply a single attention layer on the lowest resolution embedding, giving short sequence length.

6. Model: Latent Transformers

Replace each image with its latent code s and image generation can become autoregressive next-index prediction, i.e. predict $p(s_i|s_{<i})$. Then we can maximize log likelihood,

$$L_{transformer} = \mathbb{E}_{x\ p(x)}[-\log p(s)]$$

7. Applications: Conditioned Synthesis

Sometimes we want to condition on some information, like a label or a patch that primes image generation,

$$p(s|c) = \prod_i p(s_i|s_{<i}, c)$$

All we need to prime with an image is to learn a VQGAN giving us a codebook for that image, r, then prepend r to s and calculate $p(s_i|s_{<i}, r)$.

When generating whole images, need to get around sequence length limits of transformers, so generate in sliding window pattern.

8. Results

Better NLL scores than PixelSNAIL in wide variety of synthesis tasks.

10.7 FID on CelebA HQ and 11.4 on FFHQ with 10x less params than VQVAE2