Lossy Image Compression with Conditional Diffusion Models

Jiangong Yu

Master of Computer Science, Texas A&M University, College Station, TX

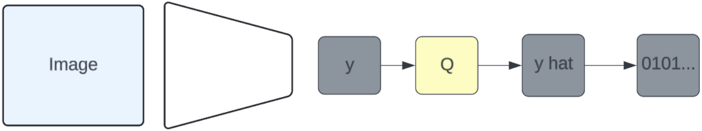
Aug 12, 2024

References

[1] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018. [Online]. Available: https://arxiv.org/abs/1802.01436

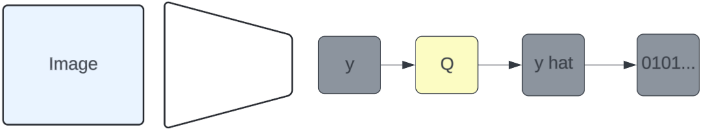
[2] R. Yang and S. Mandt, "Lossy image compression with conditional diffusion models," 2024. [Online]. Available: https://arxiv.org/abs/2209.06950

Background



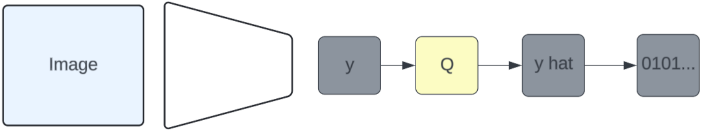
Arithmetic coding: Huffman coding...

Background



Arithmetic coding: Huffman coding...
Entropy Model: $p_{\hat{y}}(\hat{y})$
- Represented as a joint, or even fully factorized, distribution
Actual marginal distribution of the latent representation: $m(\hat{y})$

Background



Arithmetic coding: Huffman coding...
Entropy Model: $p_{\hat{y}}(\hat{y})$
- Represented as a joint, or even fully factorized, distribution
Actual marginal distribution of the latent representation: $m(\hat{y})$
- distribution of the image being encoded
- distribution of method used to infer the alternative representation y
Shannon cross entropy between the two distributions:

$$R = \mathbb{E}_{\hat{y} \sim m}[-\log_2 p_{\hat{y}}(\hat{y})]$$

Background

Side Information:

- additional bits of information sent from the encoder to the decoder
- indicate the entropy model to reduce the mismatch between the model and the actual distribution

Using **VAE** to minimize the total expected code length by learning to balance the amount of side information with the expected improvement of the entropy model.

Variational AutoEncoder

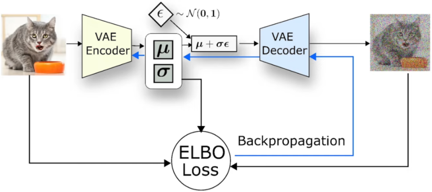


Figure: Architecture of VAE

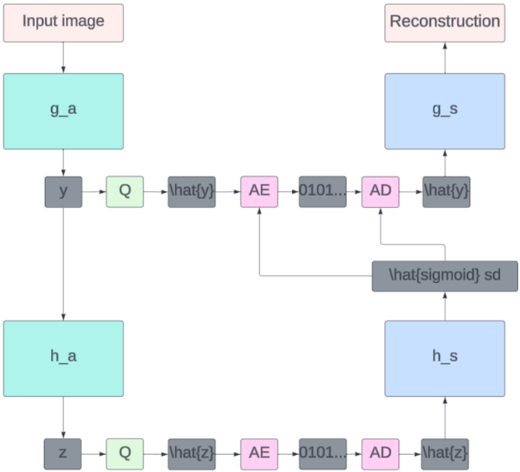
- **Reconstruction Loss:** Measuring the difference between the original input data and the data reconstructed through the VAE decoder

$$L_{recon} = \sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2$$

- **KL Divergence:** Measurement of the difference between the latent distribution of the encoder output and the a priori latent distribution (usually assumed to be the standard normal distribution)

$$Loss = L_{recon} + \beta \cdot D_{KL} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta \cdot \mathrm{KL}[q(z|x) \parallel p(z)]$$

Model Structure

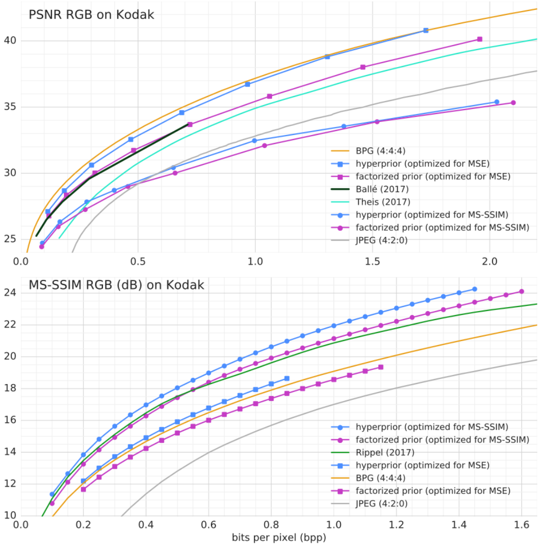


Loss function

$$Loss = \mathbb{E}_{x\sim p_x}\mathbb{E}_{\tilde{y},\tilde{z}\sim q}[-\log p_{x|\tilde{y}}(x|\tilde{y}) - \log p_{\tilde{y}|\tilde{z}}(\tilde{y}|\tilde{z}) - \log p_{\tilde{z}}(\tilde{z})]$$

- $-\log p_{x|\tilde{y}}(x|\tilde{y})$: distortion of the reconstructed image
- $-\log p_{\tilde{y}|\tilde{z}}(\tilde{y}|\tilde{z}) - \log p_{\tilde{z}}(\tilde{z})$: cross entropies encoding $\tilde{y}$ and $\tilde{z}$
- $-\log p_{\tilde{z}}(\tilde{z})$: side information

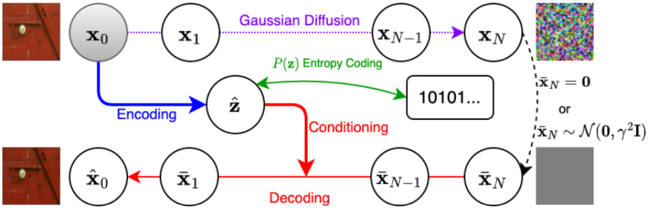
Result


PSNR RGB on Kodak
40
35
30
25
0.0
0.5
1.0
1.5
2.0
BPG (4:4:4)
hyperprior (optimized for MSE)
factorized prior (optimized for MSE)
Ballé (2017)
Theis (2017)
hyperprior (optimized for MS-SSIM)
factorized prior (optimized for MS-SSIM)
JPEG (4:2:0)
MS-SSIM RGB (dB) on Kodak
24
22
20
18
16
14
12
10
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
bits per pixel (bpp)
hyperprior (optimized for MS-SSIM)
factorized prior (optimized for MS-SSIM)
Rippel (2017)
hyperprior (optimized for MSE)
BPG (4:4:4)
factorized prior (optimized for MSE)
JPEG (4:2:0)

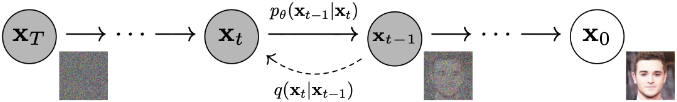
1. Variational image compression with a scale hyperprior
   - Background
   - Variational AutoEncoder
   - Structure
   - Loss function
   - Result

2. **Lossy Image Compression with Conditional Diffusion Models**
   - Denoise Diffusion Model
   - Model Structure
   - Loss
   - Decode Process
   - Result

Lossy Image Compression with Conditional Diffusion Models



Denoise Diffusion Model



$$q(x_n|x_{n-1} = \mathcal{N}(x_n|\sqrt{1-\beta_n}x_{n-1}, \beta_n\mathbf{I});$$

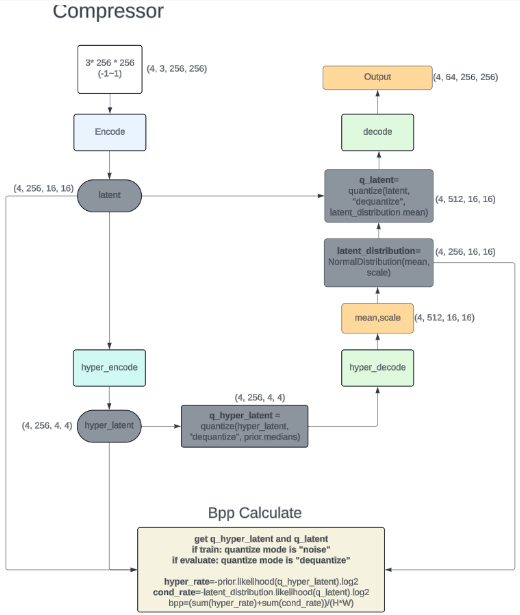
$$p_\theta(x_{n-1}|x_n) = \mathcal{N}(x_{n-1}|M_\theta(x_n, n), \beta_n\mathbf{I})$$

$\beta_n \in (0,1)$

$$L(\theta, x_0) = \mathbb{E}_{n,\epsilon}\left\|\epsilon - \epsilon_\theta(x_n(x_0), n)\right\|^2$$

- $n \sim Unif\{1, ..., N\}$
- $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
- $x_n(x_0) = \sqrt{\alpha_n}x_0 + \sqrt{1-\alpha_n}\epsilon$
- $\alpha_n = \prod_{i=1}^n(1-\beta_i)$

Encoder


Compressor
3* 256 * 256
(-1~1)
(4, 3, 256, 256)
Encode
latent
(4, 256, 16, 16)
hyper_encode
hyper_latent
(4, 256, 4, 4)
(4, 256, 4, 4)
q_hyper_latent = quantize(hyper_latent, "dequantize", prior.medians)
hyper_decode
mean,scale
(4, 512, 16, 16)
latent_distribution= NormalDistribution(mean, scale)
(4, 256, 16, 16)
q_latent= quantize(latent, "dequantize", latent_distribution.mean)
(4, 512, 16, 16)
decode
Output
(4, 64, 256, 256)
Bpp Calculate
get q_hyper_latent and q_latent
if train: quantize mode is "noise"
if evaluate: quantize mode is "dequantize"
hyper_rate=-prior.likelihood(q_hyper_latent).log2
cond_rate=-latent_distribution.likelihood(q_latent).log2
bpp=(sum(hyper_rate)+sum(cond_rate))/(H*W)

Loss

$$\mathbb{E}_{z\sim e(z|x_0)}[-\log p(x_0|z) - \lambda \log p(z)] \leq \mathbb{E}_{z\sim e(z|x_0)}[L_{upper}(x_0|z) - \lambda \log p(z)]$$

$$L_{upper}(x_0|z) = -\mathbb{E}_{x_{1:N}\sim q(x_{1:N}|x_0)}[\log \frac{p(x_{0:N}|z)}{q(x_{1:N}|x_0)}]$$

Loss

$$L_{upper}(x_0|z) \approx \mathbb{E}_{x_0,n,\epsilon} \left\| \epsilon - \epsilon_\theta(x_n, z, \frac{n}{N_{train}}) \right\|^2$$

$$L_{upper}(x_0|z) \approx \mathbb{E}_{x_0,n,\epsilon} \frac{\alpha_n}{1-\alpha_n} \left\| x_0 - \chi_\theta(x_n, z, \frac{n}{N_{train}}) \right\|^2$$

$$\epsilon(x_n, z, \frac{n}{N}) = \frac{x_n - \sqrt{\alpha_n}\chi_\theta(x_n, z, \frac{n}{N})}{\sqrt{1-\alpha_n}}$$

Loss

$$\mathbb{E}_{z\sim e(z|x_0)}[-\log p(x_0|z) - \lambda \log p(z)] \leq \mathbb{E}_{z\sim e(z|x_0)}[L_{upper}(x_0|z) - \lambda \log p(z)]$$

$$L_{upper}(x_0|z) \approx \mathbb{E}_{x_0,n,\epsilon} \frac{\alpha_n}{1-\alpha_n} \left\| x_0 - \chi_\theta(x_n, z, \frac{n}{N_{train}}) \right\|^2$$

$$L = \mathbb{E}_{z\sim e(z|x_0)}[L_{upper}(x_0|z) - \lambda \log p(z)]$$

Optional Perceptual Loss

LPIPS(Learned Perceptual Image Patch Similarity):

1. Extract features from images(using VGG, or AlexNet)
2. Compute the differences across various feature maps(Using Euclidean distance)
3. Weighted summation

More closely approximates human visual perception, especially effective in handling complex textures and fine details.

$$L_p = \mathbb{E}_{\epsilon,n,z\sim e(z|x_0)}[d(\bar{x}_0, x_0)]$$

Optional Perceptual Loss

$$L_p = \mathbb{E}_{\epsilon,n,z\sim e(z|x_0)}[d(\bar{x}_0, x_0]$$

$$L_c = \mathbb{E}_{z\sim e(z|x_0)}[L_{upper}(x_0|z) - \frac{\lambda}{1-\rho} \log p(z)]$$

$$L = \rho L_p + (1-\rho)L_c$$

$\rho \in [0,1)$: trade-off between bitrate, distortion, and perceptual quality.

Decode Process (diffusion process)

After get z from the compressor model
Init the start image:

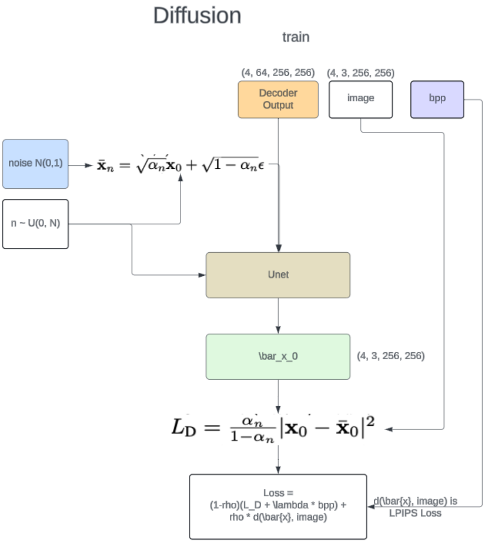
- Deterministic: $x_N = 0$
- Stochastic: $x_N \sim \mathcal{N}(0, \gamma^2 I)$

DDIM:

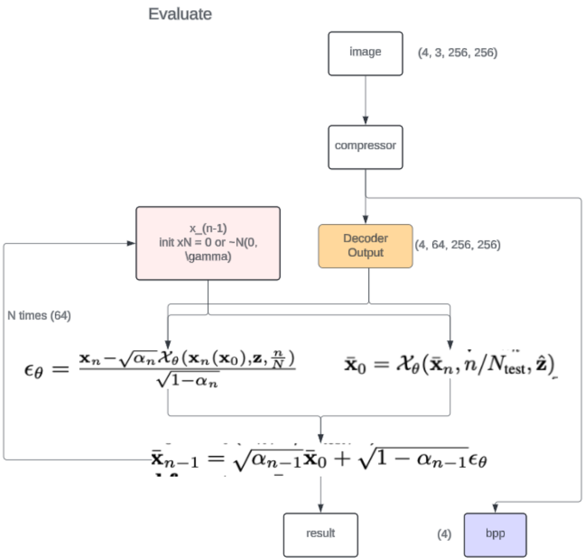
$$x_{n-1} = \sqrt{\alpha_{n-1}}\chi_\theta(x_n, z, \frac{n}{N}) + \sqrt{1-\alpha_{n-1}}\epsilon_\theta(x_n, z, \frac{n}{N})$$

- χ_θ: image predict model (Unet)
- $\beta_n \in (0, 1)$: variance schedule
- $\alpha_n = \prod_{i=1}^{n}(1-\beta_i)$
- $\epsilon(x_n, z, \frac{n}{N}) = \frac{x_n - \sqrt{\alpha_n}\chi_\theta(x_n, z, \frac{n}{N})}{\sqrt{1-\alpha_n}}$

Decode process structure: Train



Decode process structure: Evaluate



Two different types of metrics

- **Perceptual Metrics:** These are mainly used to evaluate how the visual quality of an image or video is perceived by the human eye. e.g. FID, LPIPS
- **Distortion Metrics:** These metrics are primarily used to objectively measure the technical quality of an image or video by calculating the difference between the original and processed images to assess the degree of distortion. e.g. MS-SSIM, PSNR

Weighted file compare

$\gamma = 0.8,\ \rho = 0,\ \lambda = 0.0032$

ground truth

Reproduction
$bpp = 0.8786$
$\uparrow PSNR = 78.3296$
$\downarrow LPIPS = 0.1553$

Original
$bpp = 0.7661$
$\uparrow PSNR = 76.5053$
$\downarrow LPIPS = 0.1485$

Result: Reproduction

$\gamma = 0.8,\ \rho = 0,\ \lambda = 0.0032$

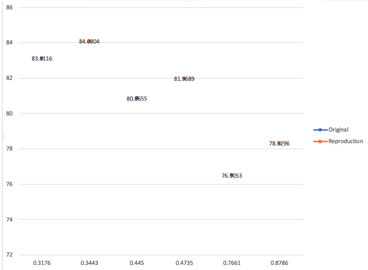
$bpp = 0.8786$
$\uparrow PSNR = 78.3296$
$\downarrow LPIPS = 0.1553$

$bpp = 0.3443$
$\uparrow PSNR = 84.0804$
$\downarrow LPIPS = 0.2313$

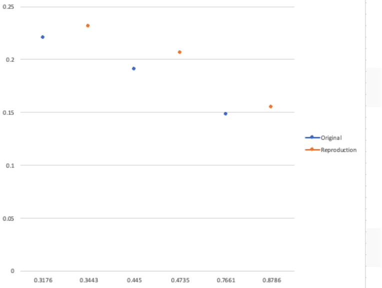
$bpp = 0.4735$
$\uparrow PSNR = 81.9689$
$\downarrow LPIPS = 0.2063$

Result

$\gamma = 0.8,\ \rho = 0,\ \lambda = 0.0032$



BPP VS PSNR



BPP VS LPIPS

Result

Different γ, bpp is same

$\gamma = 0$
$\uparrow PSNR = 78.3296$
$\downarrow LPIPS = 0.1553$

$\gamma = 0.6$
$\uparrow PSNR = 77.6171$
$\downarrow LPIPS = 0.143$

$\gamma = 0.8$
$\uparrow PSNR = 77.0089$
$\downarrow LPIPS = 0.1342$

$\gamma = 1$
$\uparrow PSNR = 76.1467$
$\downarrow LPIPS = 0.1305$

Thank You!