Practical Full Resolution Learned Lossless Image Compression

최 주 성
Vision and Display System Lab.
Sogang University

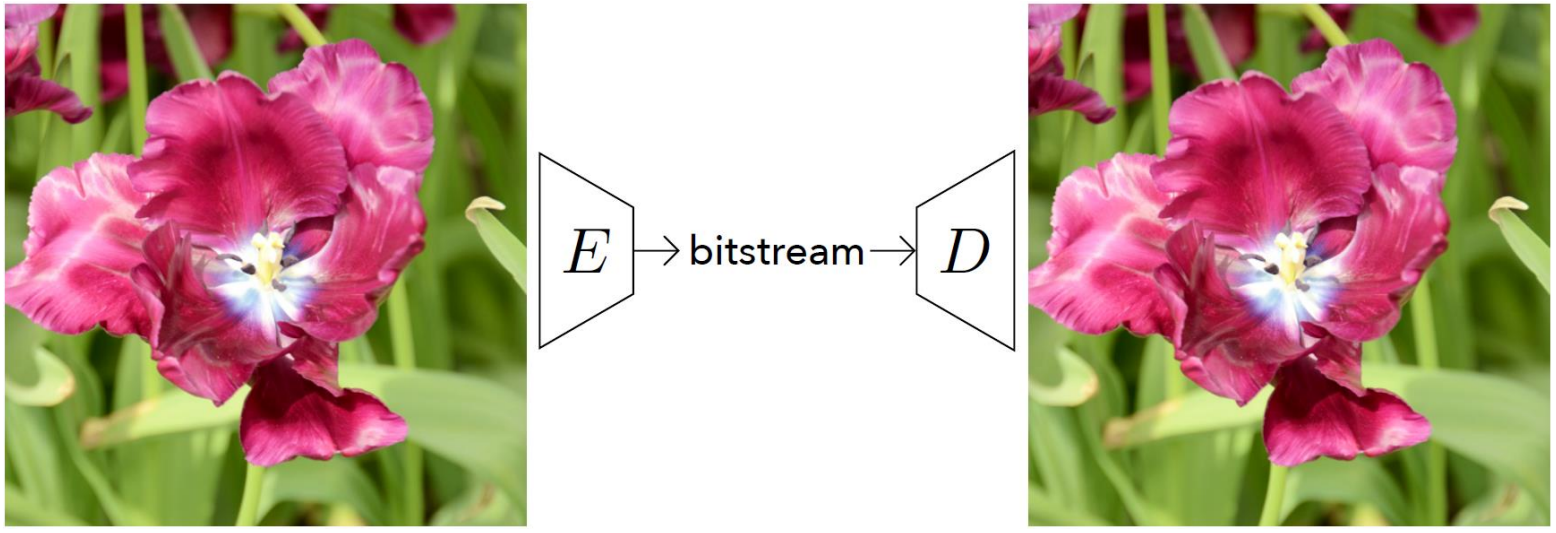
Outline

- Introduction
- Related works
- Proposed method
- Experimental results
- References

Introduction

- Image Compression Using Deep Learning
  - Autoregressive
    - 자기 자신을 입력으로 하여 자기 자신을 예측하는 모델
    - 이전 상태의 정보들에 기반하여 조건부 예측을 수행
    - 텍스트 , 오디오 , 이미지 등의 다양한 종류의 데이터의 특성을 반영하기에 좋은 구조

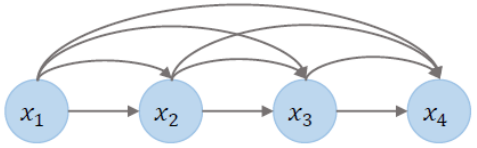
      ⁘ Necessary Math → The Chain Rule for Probabilities



Introduction

- Image Compression Using Deep Learning
  - Autoregressive
    - 자기 자신을 입력으로 하여 자기 자신을 예측하는 모델
    - 이전 상태의 정보들에 기반하여 조건부 예측을 수행
    - 텍스트 , 오디오 , 이미지 등의 다양한 종류의 데이터의 특성을 반영하기에 좋은 구조

Necessary Math → The Chain Rule for Probabilities



$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t|x_1, \ldots, x_{t-1})$$

$$p(x_1, x_2, x_3, x_4) = \boldsymbol{p(x_4|x_1, x_2, x_3)}\, p(x_1, x_2, x_3)$$

make	a	new	image
make	a	new	image
make	a	new	image
make	a	new	image

Joint probability를
연속된 conditional probability 로 분해

$p(x_1)$

$p(x_2|x_1)$

$p(x_3|x_1, x_2)$

$p(x_4|x_1, x_2, x_3)$

Introduction

- Image Compression Using Deep Learning
  - Autoregressive
    - Advantages and Disadvantages

장점

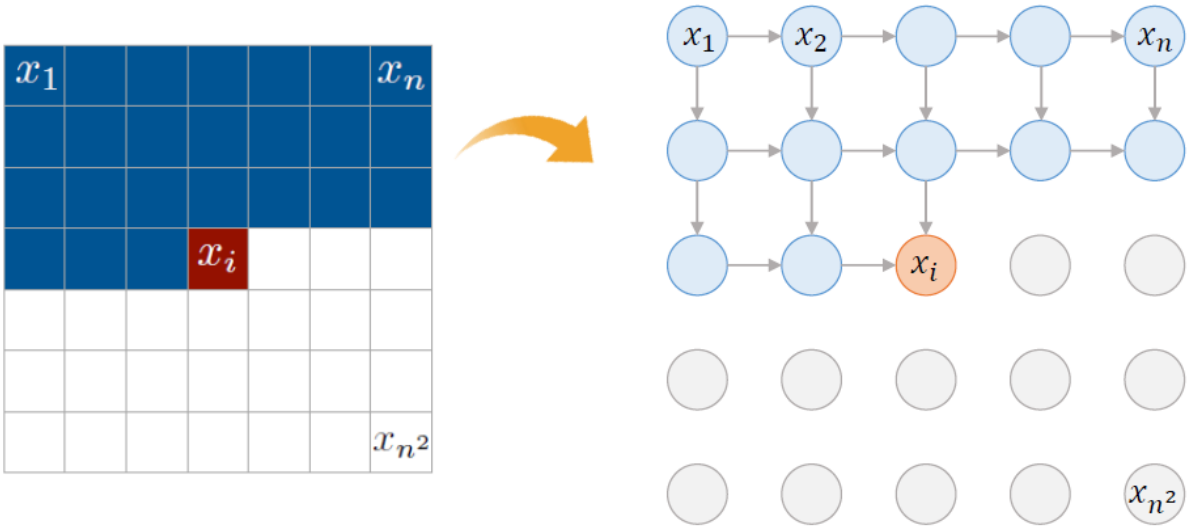
1. 정의하기 쉬움
   - ✓ Data sequence의 순서를 정의가 쉬움
2. Generation 과정이 단순함
   - ✓ Step마다 예측된 결과물에서 sample을 추출하여 그 다음 step에 새로운 input으로 넣어주는 것을 반복함
3. 이미지, 오디오, 비디오에서 log-likelihood 결과가 좋음

단점

1. 순서를 정의하는 방법에 dependent함
   - ✓ 학습 뿐만 아니라 generation 과정에서 예측 값이 input 으로 사용되기 때문
2. Generation 과정이 오래 걸림
   - ✓ 학습 과정은 병렬화가 가능하지만, sequential generation으로 특성상 느림

Related works

- Pixel RNN
  - 이미지의 픽셀 하나하나를 일련의 sequence로 생각
  - Autoregressive modeling 을 수행

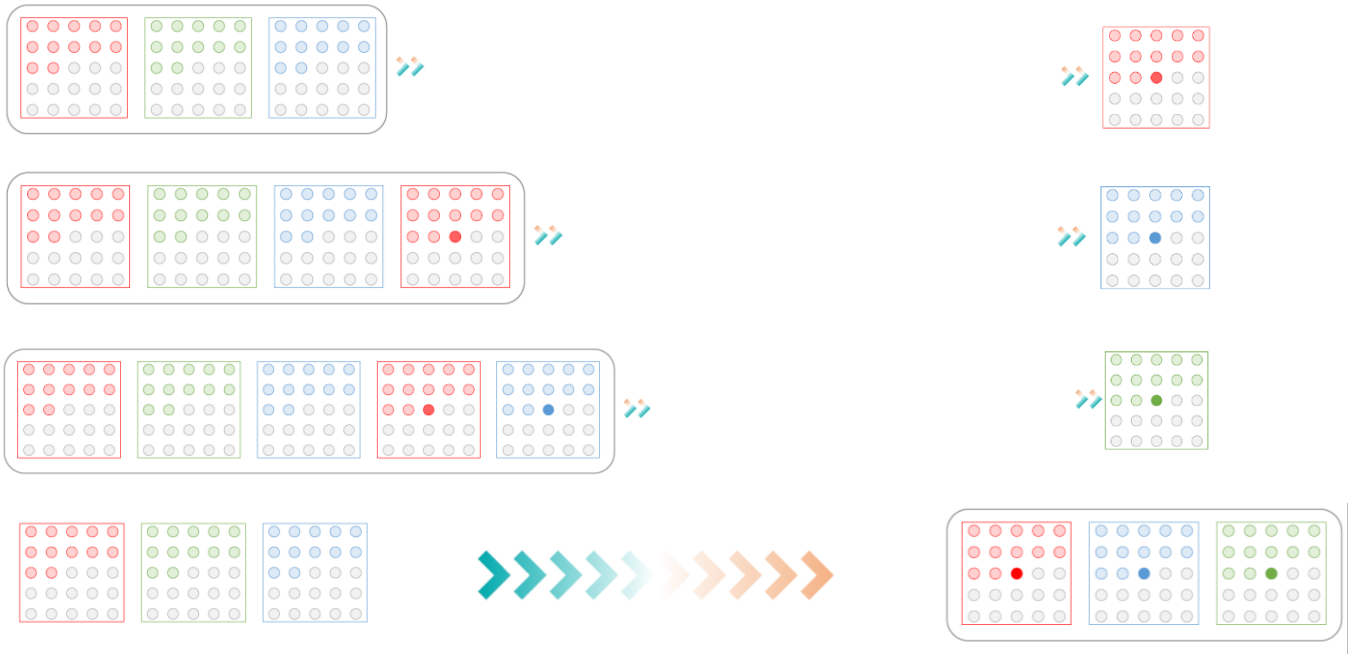


1. 이미지에 있는 임의의 픽셀은 이전 픽셀들로부터 영향을 받는다
2. 이전 픽셀들이란, 왼쪽 위를 의미 (이미지의 특성을 반영)

Related works

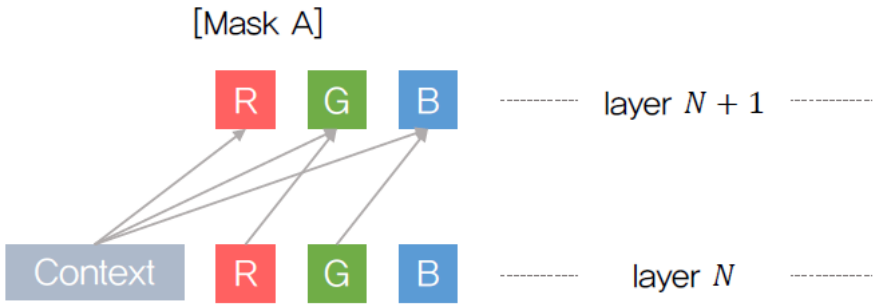
- Pixel RNN
  - 이미지 채널 간의 순서
    - Red → Green → Blue 순서로 정의

$$p(x_i|X_{<i}) = p(x_{i,R}|X_{<i})p(x_{i,G}|X_{<i}, x_{i,R})\ p\left(x_{i,B}\middle|X_{<i}, x_{i,R}, x_{i,G}\right)$$

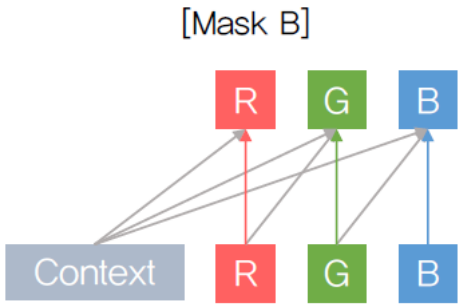


Related works

- Pixel RNN
  - Masking
    - 현재 픽셀을 생성하는데 있어서 현재를 포함한 미래의 정보를 사용하지 않아야 함
    - 어떻게 masking 을 사용하는가에 따라서 모델링에 사용할 수 있는 정보의 범위가 달라짐



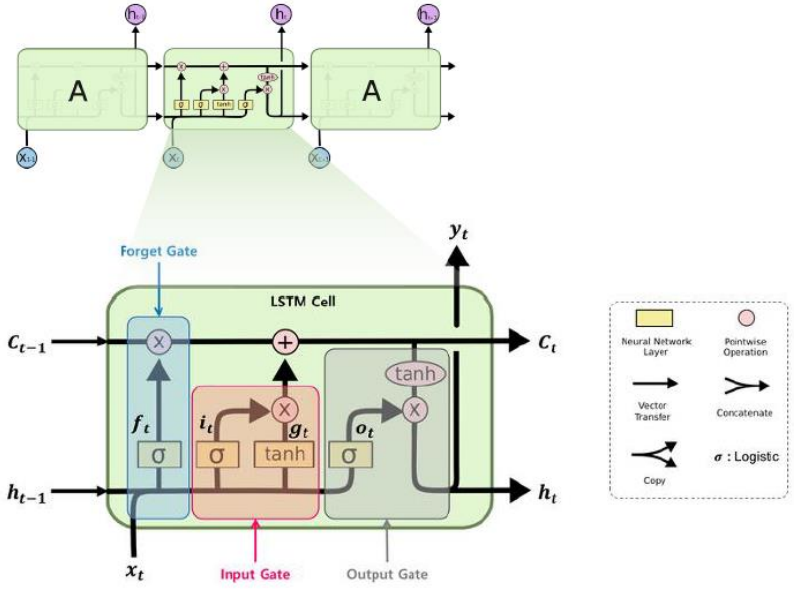
- Context는 $p(x_i|X_{<i})$를 의미
- 현재 픽셀 정보값을 사용하지 않는 masking
  * channels are not connected to themselves
- Input 이미지에 직접적으로 사용하는 방법
- 처음에 현재 자기 자신의 정보를 사용해서 자기 자신을 infer하는 것은 불가능

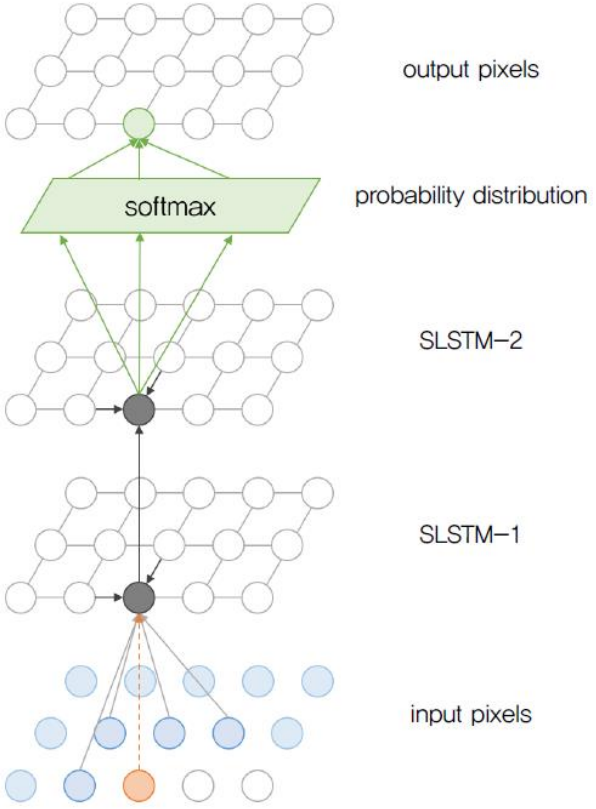


- 현재 픽셀 정보값을 사용하는 않는 masking
- Mask A를 사용한 이후에 사용
- Mask A를 통해 이미 이전 픽셀들간의 dependency를 제거했기 때문에 사용이 가능

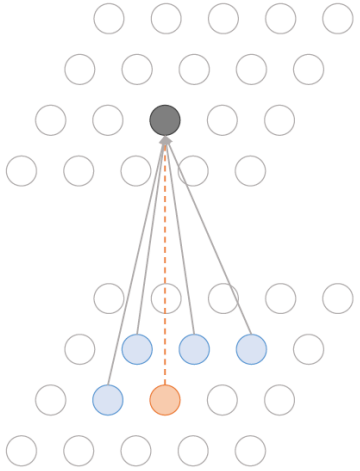
Related works

- Pixel RNN
  - Mapping architectures
    - Input to state & state to state mapping 을 따로 사용
    - LSTM 과 같이 이전 정보를 모두 사용할 수 있도록 디자인
      - Generative Image Modeling Using Spatial LSTMs (2015)





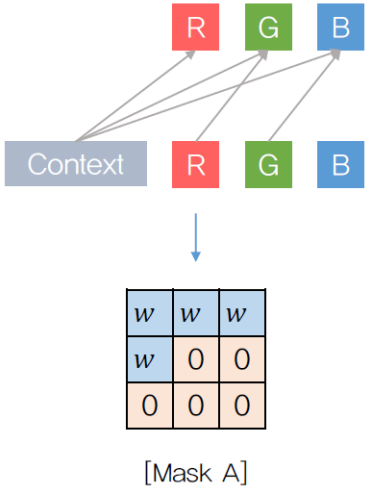
Related works

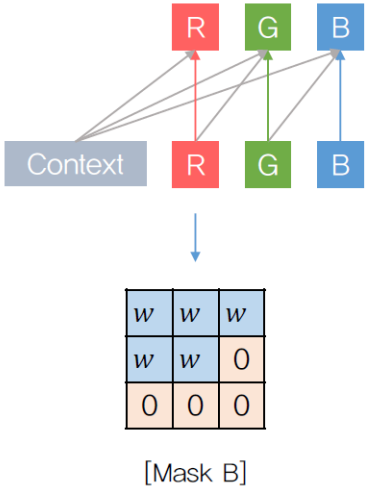


- Pixel RNN
  - Mapping architectures
    - Spatial LSTM 과 같은 예전 방법들은 , state to state 계산을 모두 수행
      - Computational cost가 너무 많으며 연산이 너무 느림
    - Computational cost 를 줄이기 위해 CNN 방법을 제안
      - Convolution 연산만을 사용해서 모델을 구축

PixelCNN

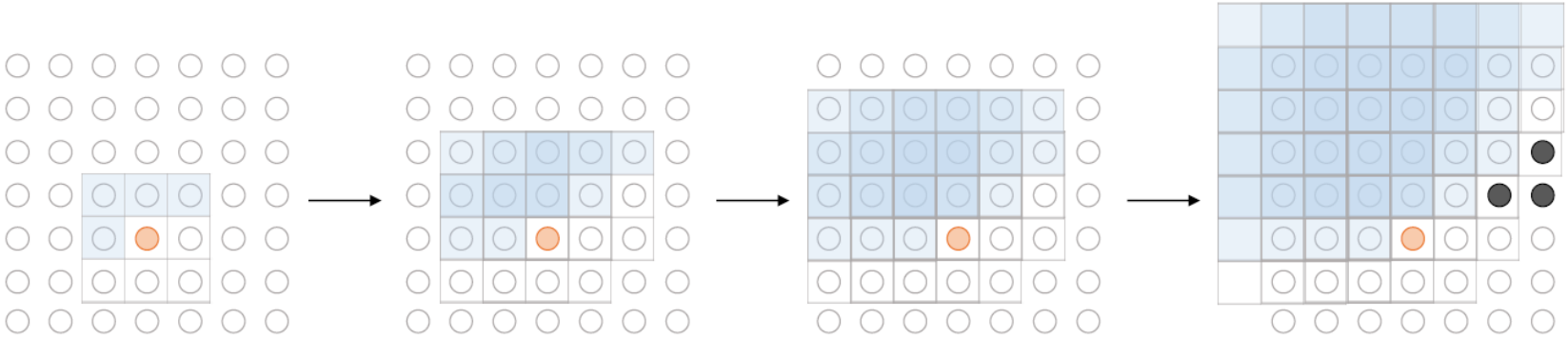
- Masked Convolution을 사용
- 미래(오른쪽 아래) 정보를 반영하지 않기 위한 트릭
- Convolution 연산이 수행되기 전에 반영하지 않을 영역에 0을 곱해서 제외



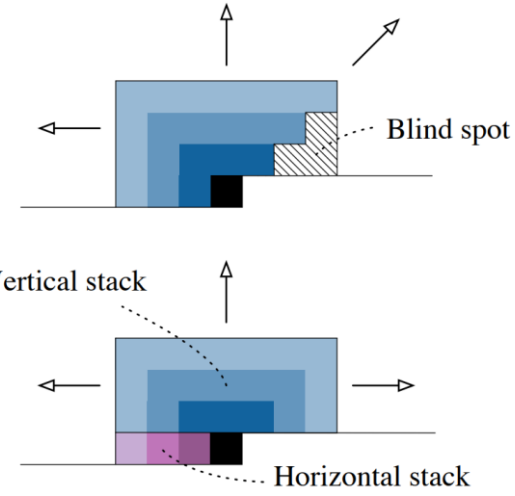


Related works

- Pixel CNN
  - 계산속도는 빠르지만, 성능이 Pixel RNN에 비해서 낮음
    - 이미지 생성 자체는 픽셀 단위에서 sequential 하게 진행
    - Pixel CNN 의 receptive field 에는 blind spot 이 존재



Related works

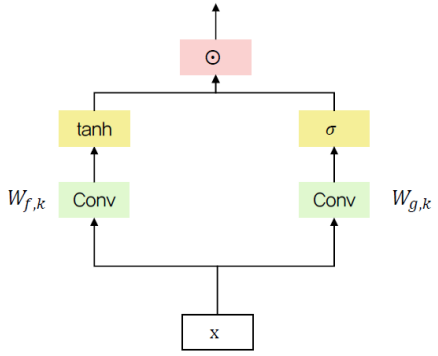


- Gated Pixel CNN
  - Horizontal & Vertical Stack
    - -Vertical
      - 현재 픽셀 위에 있는 모든 행에 대한 정보를 가져옴
      - Horizontal Stack 에 추가적인 정보로 다시 들어 감
    - -Horizontal
      - 현재 행에 대한 정보를 가져옴
        - ✓Blind Spot 을 없애고 Pixel RNN 과 같이 이전 모든 픽셀의 정보를 가져올 수 있음
  - Gated Activation
    - -Pixel RNN 의 LSTM cell 안에 사용되는 Gated Activation을 사용하도록 디자인

Gated Activation

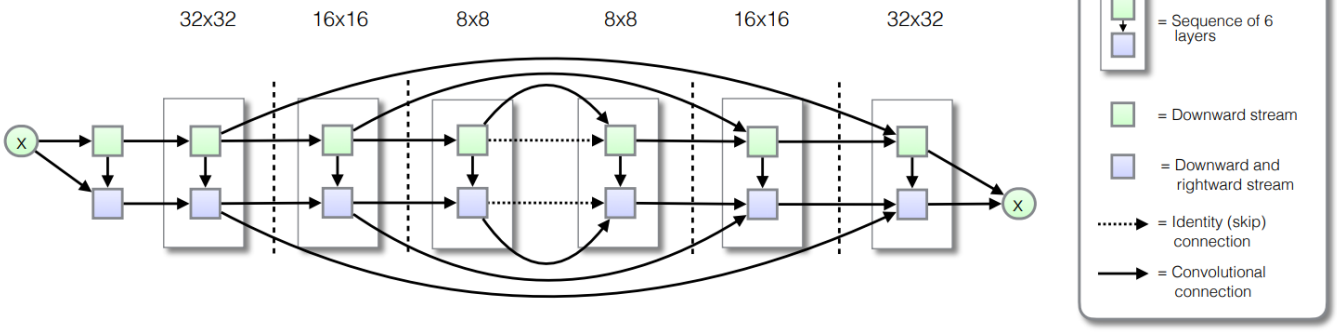
$$z = \tanh(W_{f,k} * \mathrm{x}) \odot \sigma(W_{g,k} * \mathrm{x})$$

- $*$: convolution operator
- $\odot$: element-wise multiplication operator
- k: layer index
- f, g: filter, gate → W is learnable convolution filter



Related works

- Pixel CNN++
  - Pixel CNN에 다양한 테크닉을 적용시켜서 모델의 성능을 향상시킴
    - Discretized logistic mixture likelihood
      - Softmax layer 대신 logistic distributions 의 합으로 이루어진 latent intensity variable 을 사용
    - Conditioning on whole pixels
      - Color channel 별로 output 을 산출하지 않고 , 채널의 joint distribution 로 산출하도록 변경
    - Down sampling
      - Long range dependency 에 취약한 부분을 보완하기 위해 stride 를 적용
    - Short cut connections
      - Encoder Decoder 구조를 사용 , symmetric 연결
    - Dropout
      - Regularization

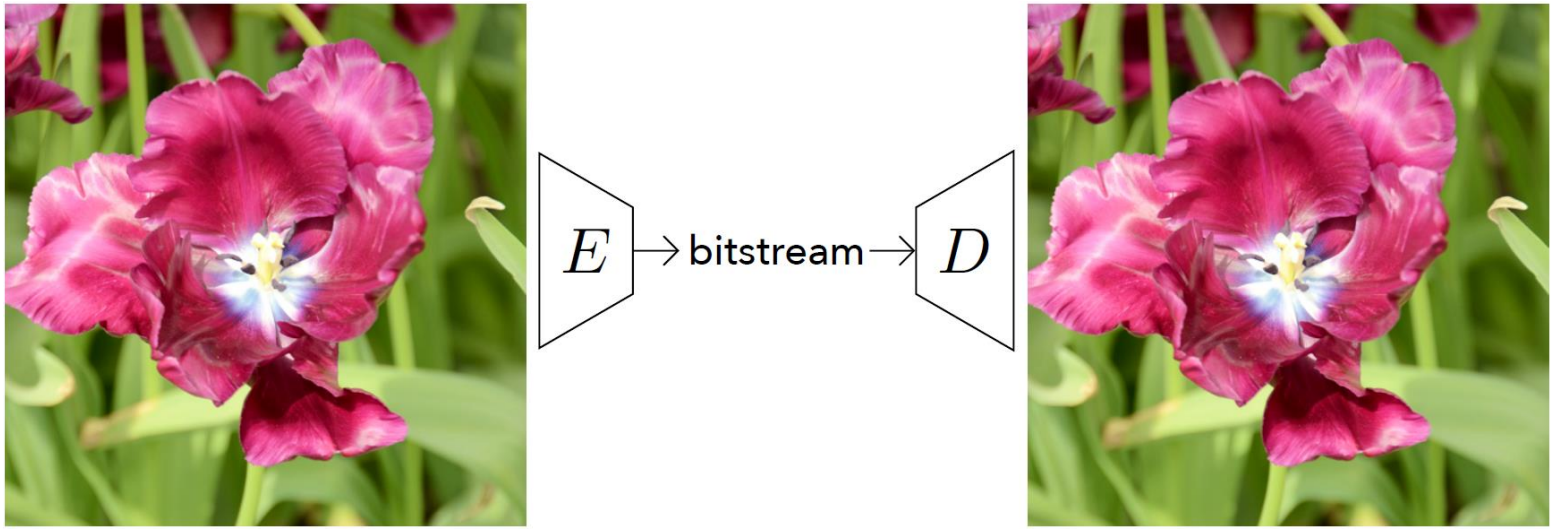


Proposed method

- Likelihood-Based Generative Models
  - Joint probability를 통한 Probability distribution modeling

-Image에 대한 정확한 확률분포 $\tilde{p}_{\text{img}}(x) = \prod_t \tilde{p}(x_t|x_{t-1}, \ldots, x_1)$ 는 알 수 없음

-Model $p($  $)$ 를 $\tilde{p}_{\text{img}}(x) = \prod_t \tilde{p}(x_t|x_{t-1}, \ldots, x_1)$ 에 minimize함



Proposed method

- Likelihood-Based Generative Models
  - Joint probability를 통한 Probability distribution modeling

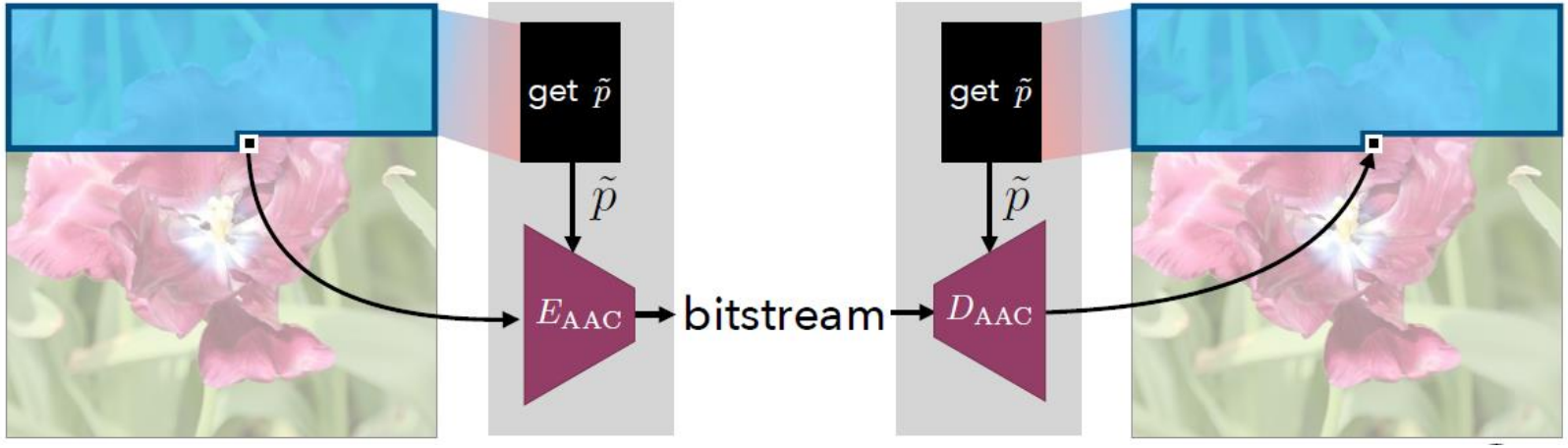
$$\tilde{p}(x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} \tilde{p}(x_i | x_1, \ldots, x_{i-1})$$

Proposed method

- Likelihood-Based Generative Models
  - Joint probability를 통한 Probability distribution modeling

$$\tilde{p}(x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} \tilde{p}(x_i | x_1, \ldots, x_{i-1})$$

Adaptive Arithmetic Coding

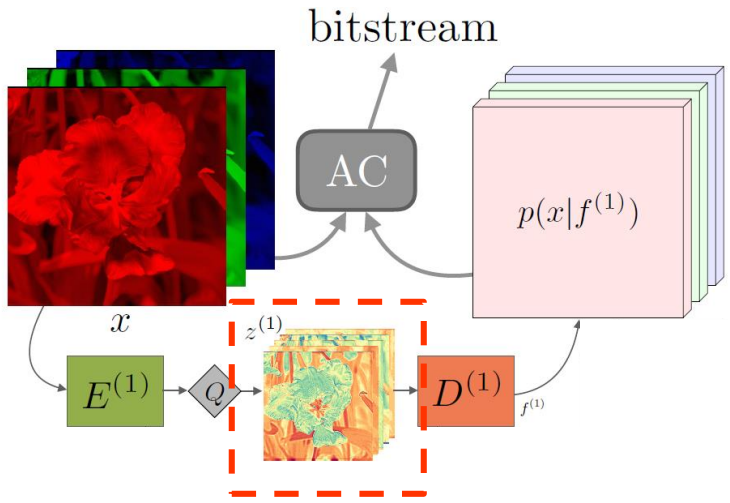


Proposed method

- Architecture
  - Modeling 작업 단순화
    - -학습 된 계층적 보조 feature map 사용

$$z^{(1)}, \ldots, z^{(S)}$$

$$p(x, z^{(1)}) =$$



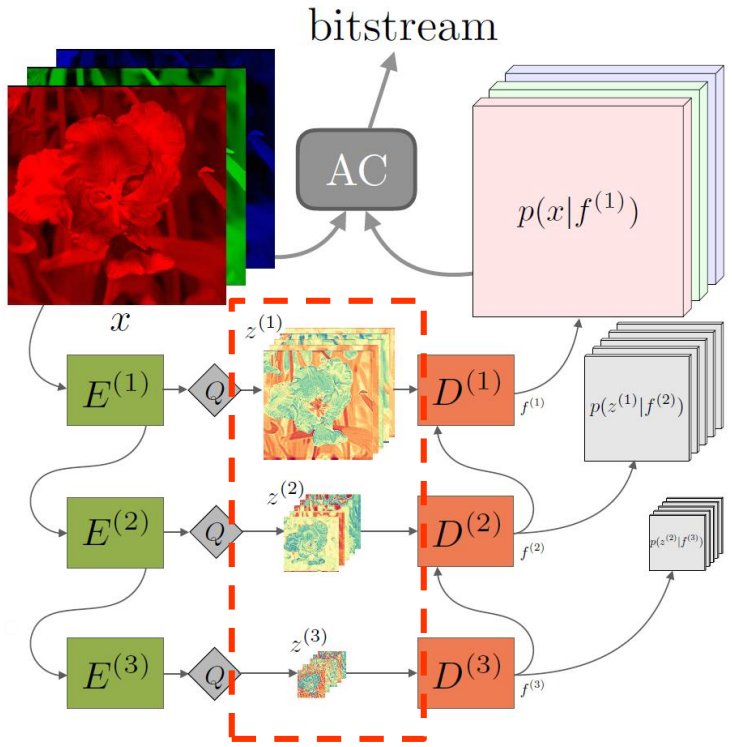
Proposed method

- Architecture
  - Modeling 작업 단순화
    - -학습 된 계층적 보조 feature map 사용

$$z^{(1)}, \ldots, z^{(S)}$$

$$p(x|z^{(1)}, \ldots, z^{(S)}) \prod_{s=1}^{S} p(z^{(s)}|z^{(s+1)}, \ldots, z^{(S)})$$

$$p(x, z^{(1)}, \ldots, z^{(S)}) = p(x\ |z_1, z_2, z_3) \cdot p(z_1|z_2, z_3) \cdot p(z_2|z_3) \cdot p(z_3)$$



Proposed method

- Architecture
  - Modeling 작업 단순화
    - –Feature extractor blocks

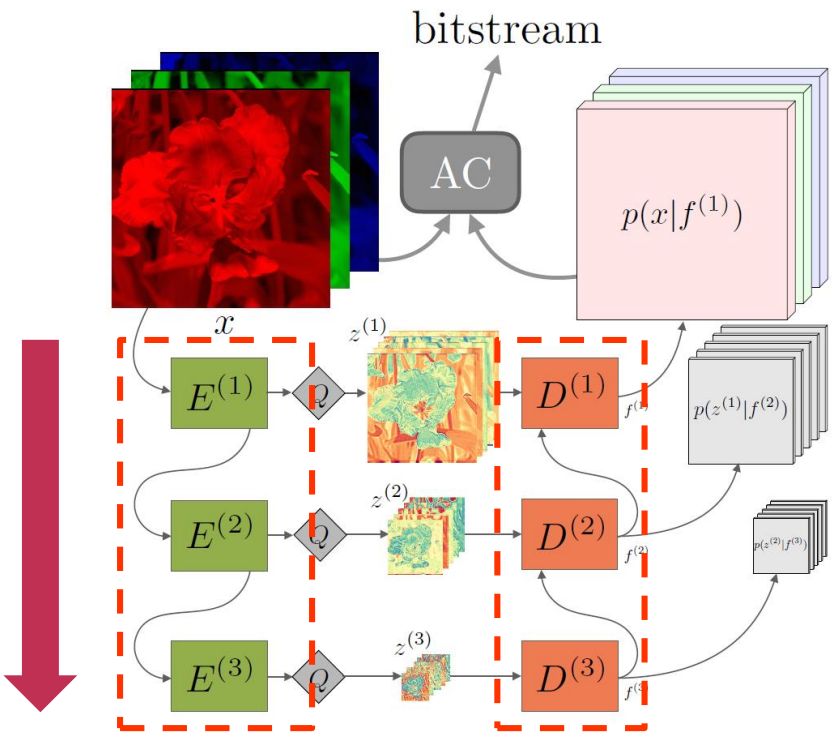
$$F^{(s)} = Q \circ E^{(s)} \circ \cdots \circ E^{(1)}$$

    - –Predictor blocks
      - EDSR [11]: super-resolution architecture

$$f^{(s)} = D^{(s)}(f^{(s+1)}, z^{(s)})$$

Bicubic subsampling operator

$$z^{(s)} = \mathcal{B}_{2^s}(x)$$
$$= F^{(s)}(x)$$



Proposed method

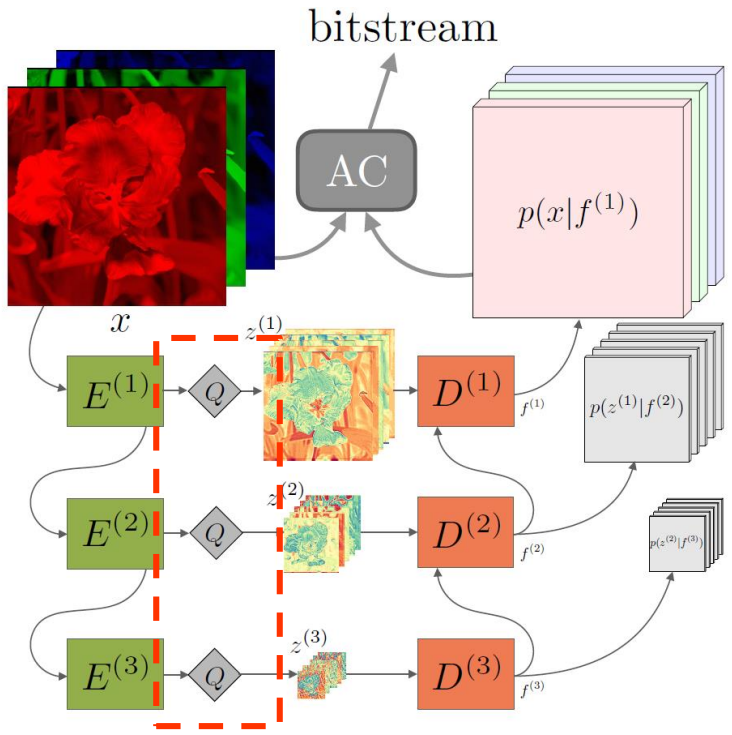
- Architecture
  - Modeling 작업 단순화
    - -Scalar differentiable quantization function [8]
      - $F^{(s)}$ 의 출력을 양자화
      - Given levels (L=25)

$$\mathbb{L} = \{\ell_1, \ldots, \ell_L\} \subset \mathbb{R}$$

$$z = Q(z') := \arg\min_j \|z' - \ell_j\|$$

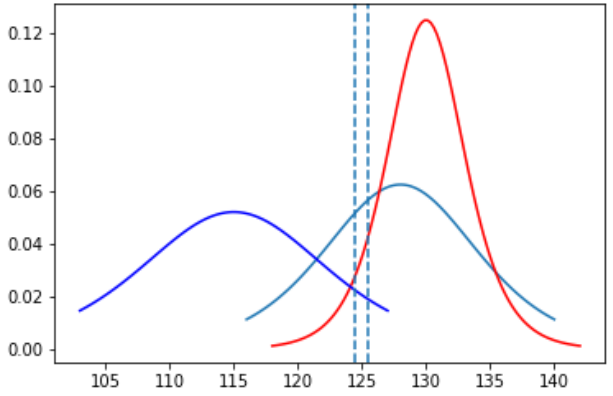
$$\tilde{Q}(z') = \sum_{j=1}^{L} \frac{\exp(-\sigma_q \|z' - \ell_j\|)}{\sum_{l=1}^{L} \exp(-\sigma_q \|z' - \ell_l\|)} \ell_j$$

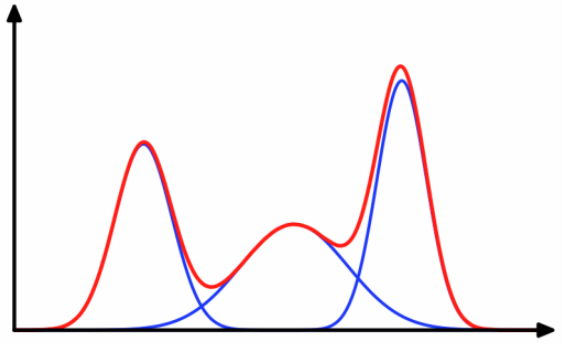
< Differentiable "soft quantization" >



Proposed method

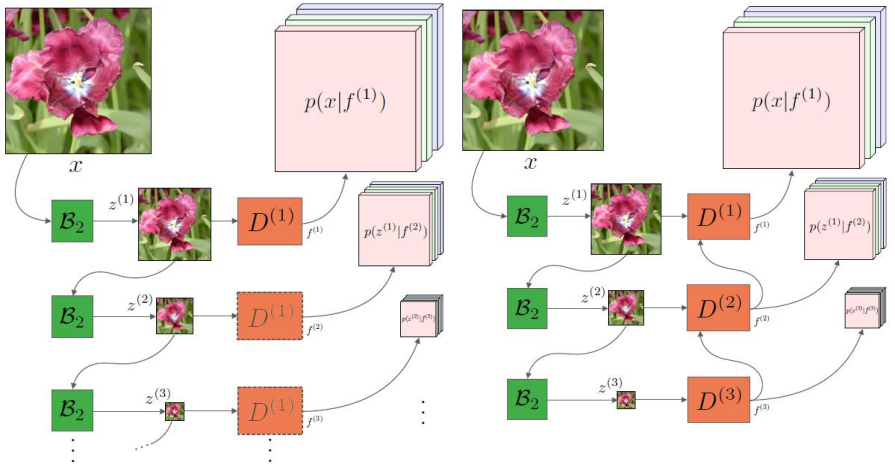
- Mixture Model
  - Discretized Logistic Mixture Likelihood
    - -Modeling의 목표
      - [0, 255]의 픽셀 값을 예측하여 이미지 생성
      - Value를 예측하기 위한 대표적인 방법으로 softrmax가 존재
    - -Softmax
      - 예측된 값들의 수학적 연관성을 무시함
        - ✓Ex. 인접 픽셀들은 존재 (color, value)
      - Training data에서 학습되지 않은 값은 test set에서 예측이 불가능
      - 평균과 분산을 갖는 distribution를 modeling 해야함
    - -Discrete values를 채택하는 방법
      - 예측된 연속한 값이 [x-0.5, x + 0.5]에 존재할 경우 x로 채택
    - -Mixture of Distributions
      - 선형 조합으로 이루어짐
      - p(x) = a_1 * p_1(x) + a_2 * p_2(x) + … a_n * p_n(x)
        - ✓a_1, a_2, …, a_n: hyperparameters





Experimental results

- Quantitative evaluation
  - Reference model for comparison
    - –RGB: feature extractors → bicubic subsampling
    - –Shared: only train one predictor



[bpsp]	Method	Open Images	DIV2K	RAISE-1k
Ours	L3C	2.991	3.094	2.387
Learned Baselines	RGB Shared	4.314 +44%	4.429 +43%	3.779 +58%
	RGB	3.298 +10%	3.418 +10%	2.572 +7.8%
Non-Learned Approaches	PNG	4.005 +34%	4.235 +37%	3.556 +49%
	JPEG2000	3.055 +2.1%	3.127 +1.1%	2.465 +3.3%
	WebP	3.047 +1.9%	3.176 +2.7%	2.461 +3.1%
	FLIF	2.867 −4.1%	2.911 −5.9%	2.084 −13%

Experimental results

- Quantitative evaluation
  - Bitcost

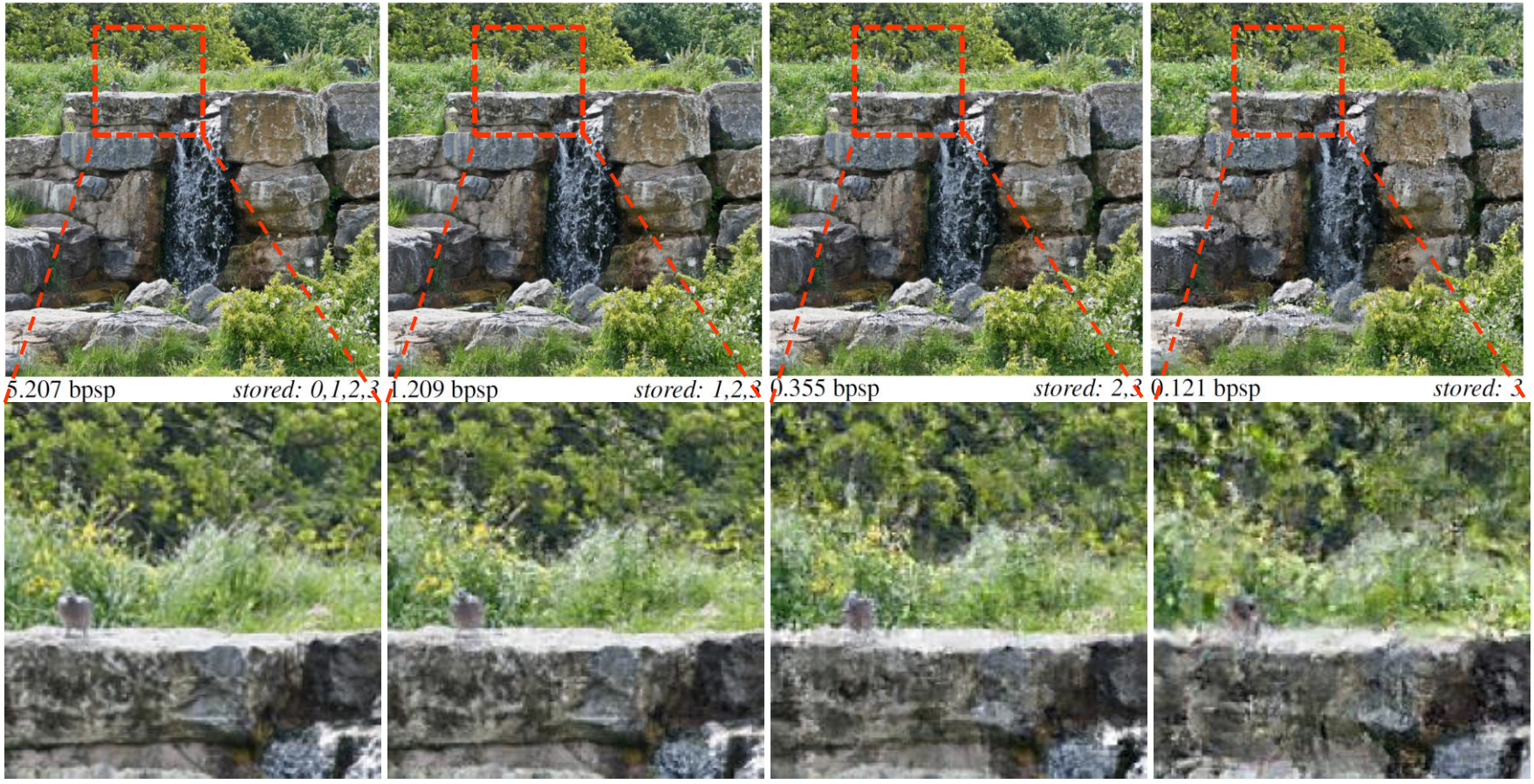
[bpsp]	ImageNet32	Learned
L3C (ours)	4.76	✓
PixelCNN [46]	3.83	✓
MS-PixelCNN [32]	3.95	✓
PNG	6.42	
JPEG2000	6.35	
WebP	5.28	
FLIF	5.08	

  - Encoding / Decoding Time

Codec	Encoding [s]	Decoding [s]	[bpsp]	GPU	CPU
L3C (Ours)	0.242	0.374	3.386	✓	✓
PNG	0.213	$6.09 \cdot 10^{-5}$	4.733		✓
JPEG2000	$1.48 \cdot 10^{-2}$	$2.26 \cdot 10^{-4}$	3.471		✓
WebP	0.157	$7.12 \cdot 10^{-2}$	3.447		✓
FLIF	1.72	0.133	3.291		✓

Experimental results

- Qualitative evaluation



References

1. A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. In ICML, 2016.
2. S. Reed, A. Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, Y. Chen, D. Belov, and N. Freitas. Parallel Multiscale Autoregressive Density Estimation. In ICML, 2017.
3. T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixel-CNN++: A PixelCNN Implementation with Discretized Logistic Mixture Likelihood and Other Modifications. ICLR,2017.
4. A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. In ICML, 2016.
5. I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. Communications of the ACM, 30(6):520–540, 1987.
6. L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. ICLR, 2016.
7. T. M. Cover and J. A. Thomas. Elements of Information Theory. John Wiley & Sons, 2012.
8. F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool. Conditional Probability Models for Deep Image Compression. In CVPR, 2018.
9. G. Hinton, N. Srivastava, and K. Swersky. Neural Networks for Machine Learning Lecture 6a Overview of mini-batch gradient Descent.
10. E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool. Soft-to-Hard Vector Quantization for End-to-End Learning Compressible Representations. In NIPS, 2017.
11. B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. In CVPR Workshops, 2017


Thank
you