Normalizing Flows

2022 하계세미나

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
JoonKyu Kim

목차

- Generative Models
  - Generative Adversarial Network (GAN)
  - Variational AutoEncoder (VAE)
  - Normalizing Flows
    - Density estimation
    - Anomaly detection
    - **Image generation**
- RealNVP
- Glow
- SRFlow

Generative Models

- GAN
  - Fake sample을 생성하는 generator와 이를 real data와 구별하는 discriminator를 학습
    - Content loss(L1, L2) 와 adversarial loss를 사용하여 학습

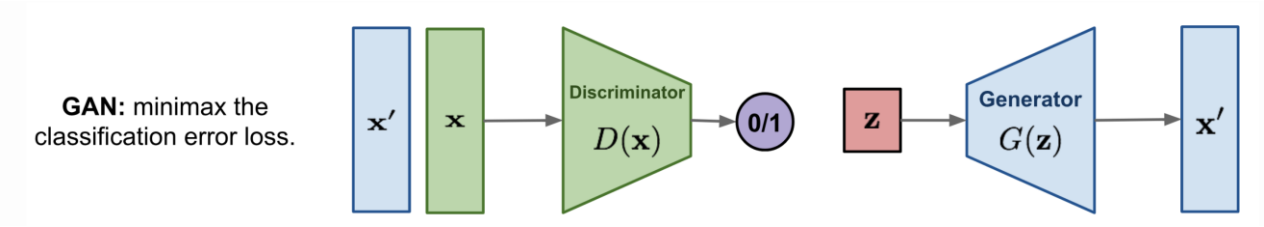


Fig.1 GAN structure

- VAE
  - Evidence lower bound(ELBO)의 maximize를 통해 log-likelihood를 최적화
  - Low dimension의 latent space가 decoder를 통해 생성한 결과에서 blurry함이 나타남

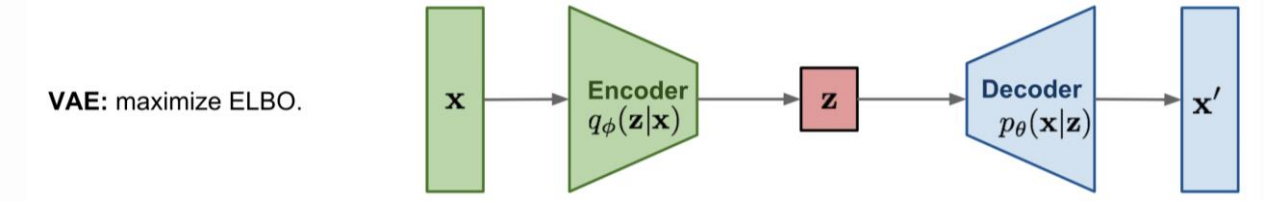


Fig.2 VAE structure

Generative Models

- Normalizing Flow
  - Invertible transformation을 사용하여 exact negative log-likelihood를 사용하여 학습
  - Invertible하므로 latent space z와 x의 dimension이 동일함

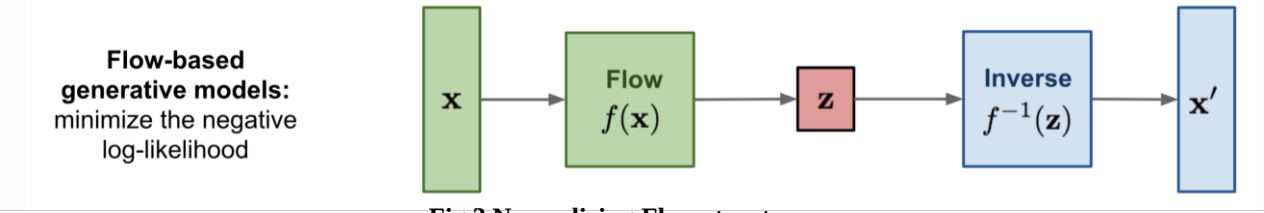


Fig.3 Normalizing Flow structure

Method	Train Speed	Sample Speed	Num. Params.	Resolution Scaling	Free-form Jacobian	Exact Density	FID	NLL (in BPD)
Generative Adversarial Networks								
DCGAN [182]	★★★★★	★★★★★	★★★★★	★★★☆☆	✓	✗	37.11	-
ProGAN [114]	★★★★☆	★★★★★	★★★★☆	★★★★★	✓	✗	15.52	-
BigGAN [19]	★★★★☆	★★★★★	★★★★★	★★★★☆	✓	✗	14.73	-
StyleGAN2 + ADA [115]	★☆☆☆☆	★★★★★	★★★☆☆	★★★★★	✓	✗	2.42	-
Variational Autoencoders								
Convolutional VAE [123]	★★★★★	★★★★★	★★★★★	★★☆☆☆	✓	(✓)	106.37	$\leq$ 4.54
Variational Lossy AE [29]	★★★☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	✗	(✓)	-	$\leq$ 2.95
VQ-VAE [184], [235]	★★☆☆☆	★★★☆☆	★☆☆☆☆	★★★★★	✗	(✓)	-	$\leq$ 4.67
VD-VAE [31]	★☆☆☆☆	★★★★★	★★★☆☆	★★★★★	✓	(✓)	-	$\leq$ 2.87
Normalizing Flows								
RealNVP [43]	★★☆☆☆	★★★★★	★★★★★	★★★☆☆	✗	✓	-	3.49
GLOW [124]	★☆☆☆☆	★★★★★	★★☆☆☆	★★★★☆	✗	✓	45.99	3.35
FFJORD [62]	★☆☆☆☆	★★★☆☆	★★★★★	★★☆☆☆	✓	(✓)	-	3.40
Residual Flow [26]	★★☆☆☆	★★★★☆	★★★★☆	★★★★☆	✓	(✓)	46.37	3.28

	1 Star	2 Stars	3 Stars	4 Stars	5 Stars
Training	>5 days	$\leq$5 days	$\leq$2 days	$\leq$1 days	$\leq\frac{1}{2}$ day
Sampling	AR	MCMC	Middle	$\leq$20 steps	1 step
Params	>120M	$\leq$120M	$\leq$60M	$\leq$30M	$\leq$10M
Resolution	<32	32	64 or 128	256 or 512	$\geq$1024

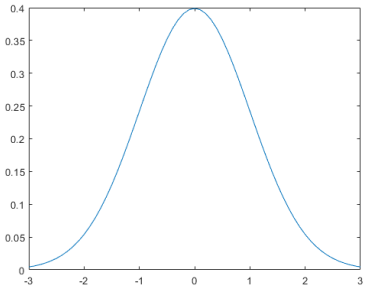
X : intractable densities
(✓) : approximate densities
✓ : tractable densities

Fig.4 Generative Models 비교

Pros and Cons

- Pros
  - 학습의 안정성
    - Generator / Discriminator의 학습 과정에서 hyperparameter 설정이 까다로운 GAN
      - Normalizing Flows는 negative log likehood loss만을 사용하여 학습하기 때문에 학습 과정이 매우 안정적임
        - ✓Exact log-likelihood를 계산할 수 있으며, 이를 사용하여 학습하기 때문에 GAN, VAE에 비해 converge하기 쉬움
  - 생성된 결과
    - 일반적으로 GAN, VAE에 비해 realistic하고 blurry하지 않은 이미지 생성 가능
- Cons
  - Bijective한 특성을 위한 구조적 제한 발생
    - Invertible해야 하기 때문에 latent space의 dimension이 input 이미지와 동일함
    - Invertible한 특성 유지하기 위해 나타나는 transformation function의 표현력 제한

Normalizing Flow



- Normal distribution
  - $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$
- Input data를 normal distribution으로 변형하기 위한 Flow들의 집합

  $f(z) = x,\ f^{-1}(x) = z$ 의 관계를 갖는 transformation 함수 : Flows

  ∴ Latent space z는 임의의 distribution이지만, computational cost를 고려하여 normal distribution으로 사용

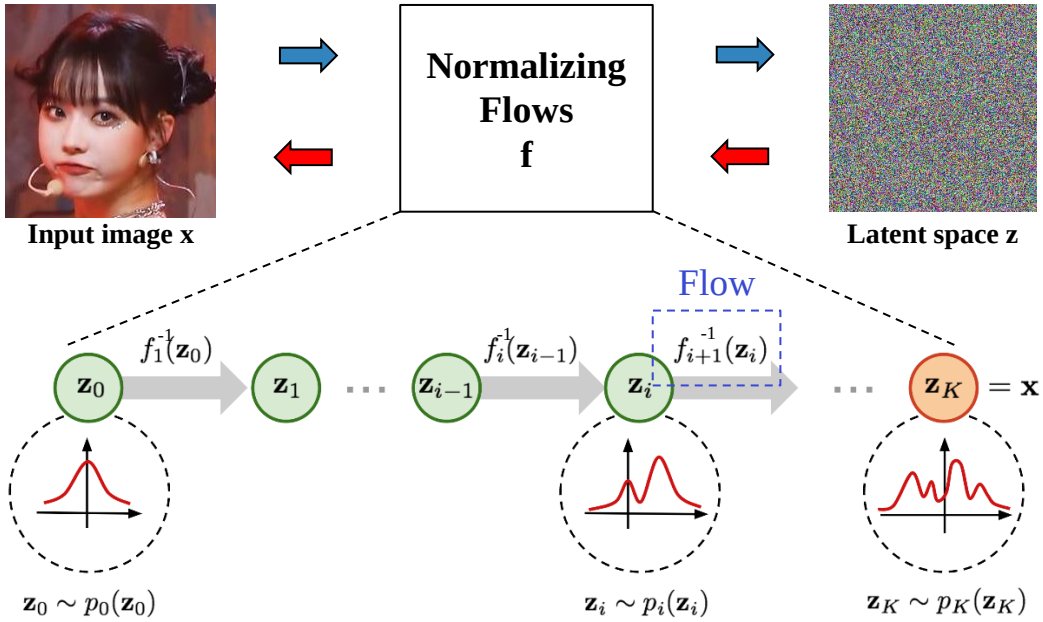


Fig.5 Normalizing Flow structure

Change of variable theorem

- Single random variable z
  - Normalizing Flows에서는 Normal distribution을 가정
- Mapping function f
  - f^{-1} 가 존재, $f^{-1}(x) = z$
- $p(x) = ?$
  - 확률분포의 정의에 따라,

$$\int p(x)dx = 1 = \int \pi(z)dz\,,$$

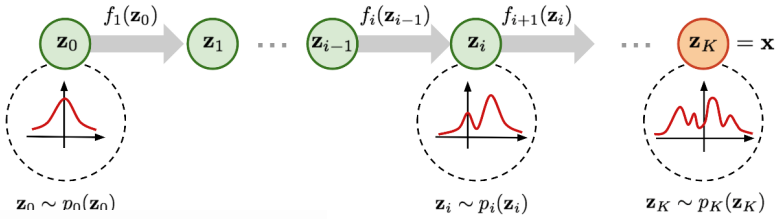
$$p(x) = \pi(z)\left|\frac{dz}{dx}\right| = \pi(f^{-1}(x))\left|\frac{df^{-1}}{dx}\right|$$

  - 다변수로 확장

Jacobian

$$p(\mathbb{x}) = \pi(\mathbb{z})\left|\frac{d\mathbb{z}}{d\mathbb{x}}\right| = \pi\left(f^{-1}(\mathbb{x})\right)\left|\frac{df^{-1}}{d\mathbb{x}}\right|$$

Change of variable theorem



$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0)$$

$$\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) = \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right|$$

$$= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right|$$

$$= \cdots$$

$$= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|$$

Normal distribution

각 layer마다 연산 가능

- p(x)를 Flow들을 통해 normal distribution으로 표현 가능
  - Transformation function이 invertible해야 함
  - 각 transformation function의 Jacobian의 연산이 쉬워야 함
    - 두 조건을 만족하는 함수로 affine transformation을 제안

RealNVP[1] [ICLR 2017]

- Affine transform

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases}$$

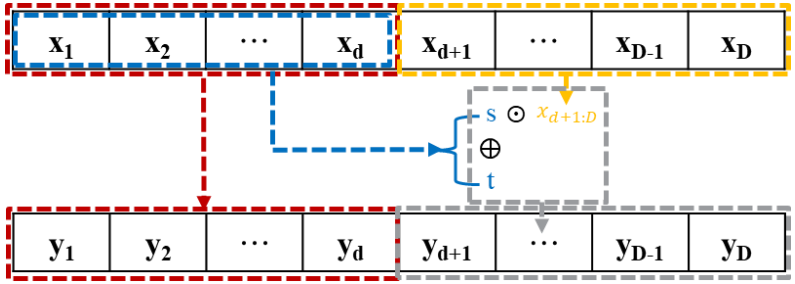


Fig.6 Affine transform structure

- Bijective
  - Weight matrix의 inverse matrix 연산 없이 inverse function을 구할 수 있음
    - ∴ 따라서 neural network로 구성된 s( ), t( ) function을 자유롭게 구성 가능
- Jacobian

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \operatorname{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix}$$

$$\det(J) = \prod_{j=1}^{D-d} \exp\left(s(x_{1:d})\right)_j = \exp\left(\sum_{j=1}^{D-d} s(x_{1:d})_j\right)$$

RealNVP[1]

- Affine transform
  - 입력 feature의 channel을 반으로 분할함 (x_1, x_2)
  - 변경하지 않는 부분 (x_1) 과 변경하는 부분(x_2)를 지정
  - x_1을 사용하여 x_2를 scale, translation하여 affine transformation을 수행

Forward propagation

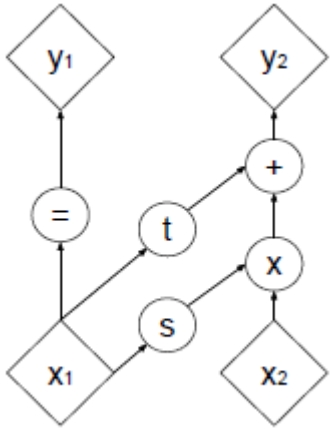
✓ $y_{1:d} = x_{1:d}$

✓ $y_{d+1:D} = x_{d+1:D} \odot e^{s(x_{1:d})} + t(x_{1:d})$

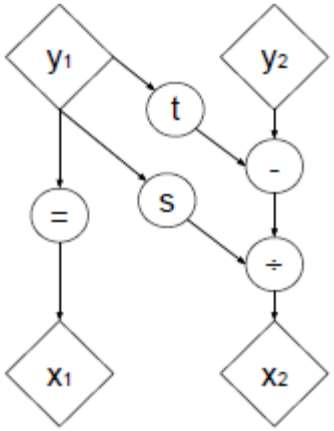
Backward propagation

✓ $x_{1:d} = y_{1:d}$

✓ $x_{d+1:D} = (y_{d+1:D} - t(y_{1:d})) \odot e^{-s(y_{1:d})}$



<Forward propagation>



<Backward propagation>

Fig.7 Affine transform structure

RealNVP[1]

- 여러 개의 coupling layer를 통해 Flows를 구성
  - $(f_b \circ f_a)(x) = z$

$$\frac{\partial (f_b \circ f_a)}{\partial x_a^T}(x_a) = \frac{\partial f_a}{\partial x_a^T}(x_a) \cdot \frac{\partial f_b}{\partial x_b^T}(x_b = f_a(x_a))$$

$$\det(A \cdot B) = \det(A)\det(B).$$

  - Inverse

$$(f_b \circ f_a)^{-1} = f_a^{-1} \circ f_b^{-1}$$

- Channel permutation

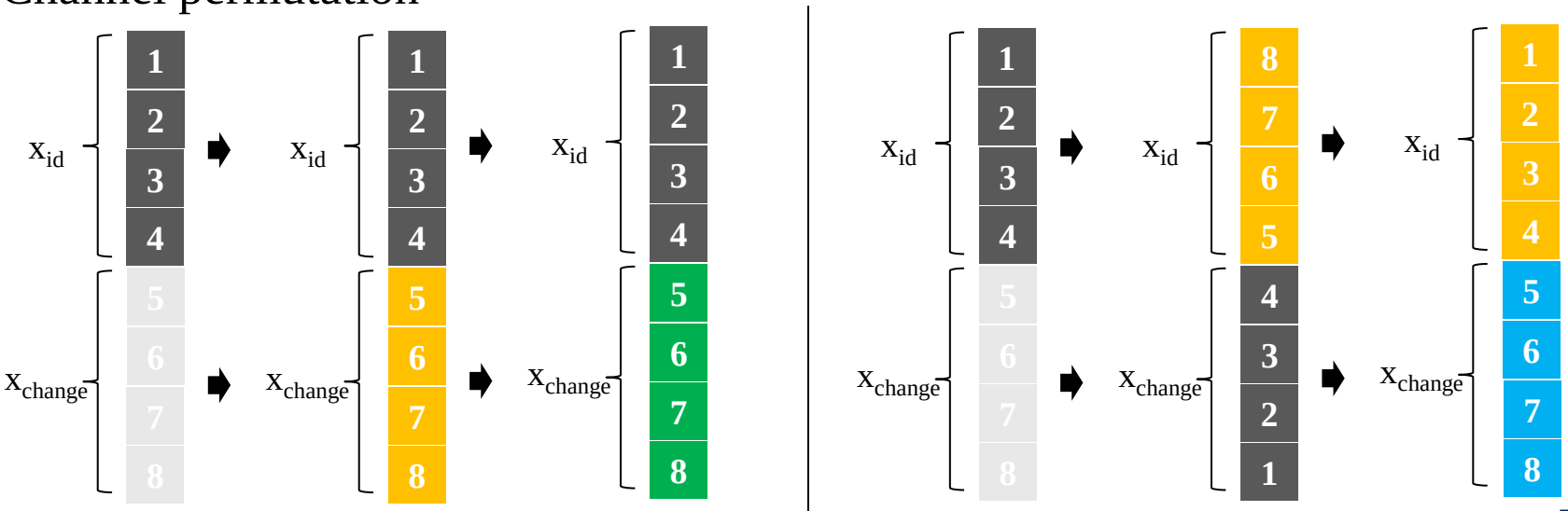


Fig.8 Squeeze operation

RealNVP[1]

- Squeezing operation
  - Channel간의 연산을 통한 affine transform을 수행해야 함
    - 이미지는 통상 RGB 3 channel로 구성됨
      - Squeezing operation을 통해 channel size를 늘림
      - Invertible한 특성을 유지하기 위해서는 C * H * W의 값은 변할 수 없음
    - 따라서 squeeze operation을 수행
      - Spatial size를 반으로 줄이는 효과
      - Shuffle 이후 reshape하여 spatial size는 ¼, channel은 4배가 됨

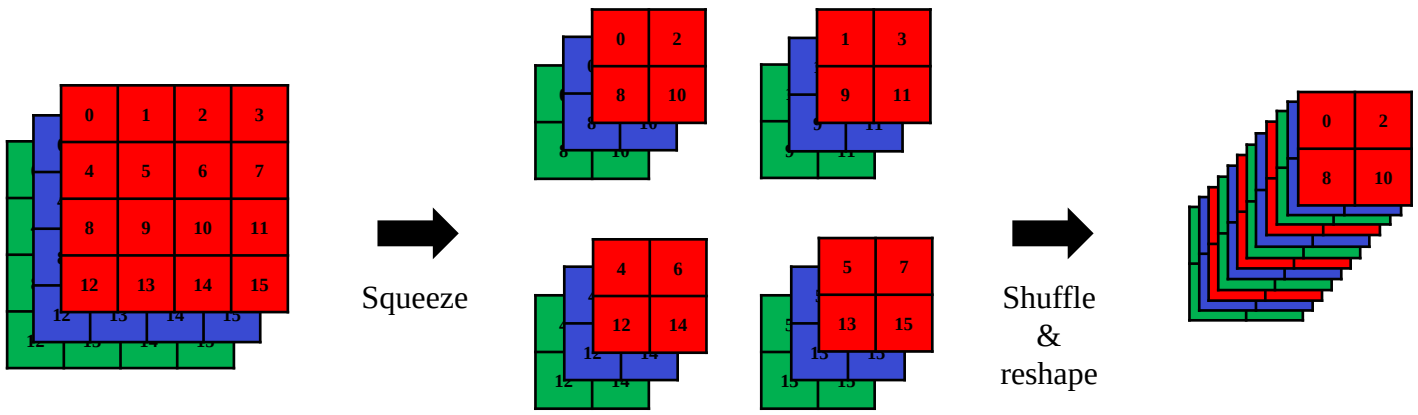
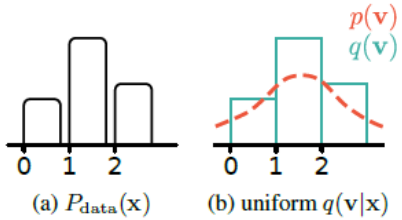


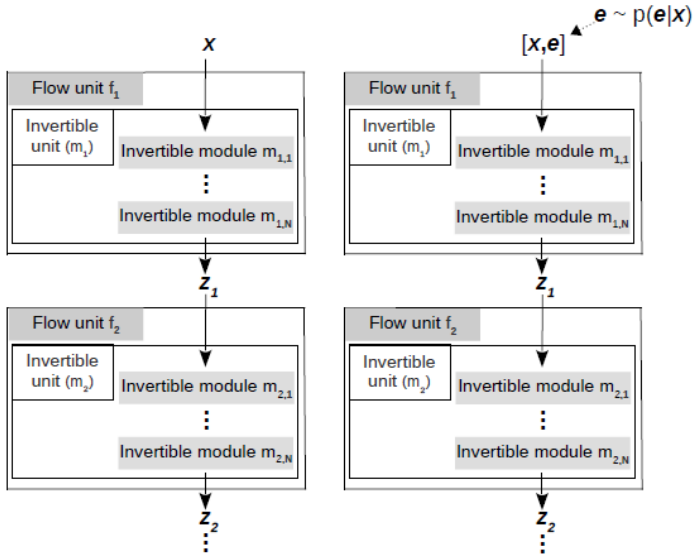
Fig.9 Squeeze operation

Dequantization

- Normalizing Flow는 확률 분포 추정 모델
  - Input image는 digital로 저장되므로 discrete함
  - Normalizing Flow는 continuous random variable의 distribution을 추정
    - 따라서 많은 dequantization 기법이 도입되어 input image를 dequantinize



- Dequantinization method
  - Adding uniform noise
    - RealNVP, Glow, …
    - Gaussian, variational, Importance-weighted, …



Glow[1] [NIPS 2018]

- RealNVP의 successor로, 3개의 모듈로 하나의 step(flow)을 구성
  - Activation normalization
    - Invertible한 channel-wise normalization을 수행
      - Forward : S * X + b
      - Backward : 1/S * (Y-b) log-det : h * w * sum(log|s|)
  - Invertible 1x1 convolution layer
    - RealNVP에서 적용한 단순하게 channel의 순서를 reverse 방향으로 바꾸는 것은 학습이 불가능한 부분이며 편향이 존재함
    - Glow에서는 이를 1x1 conv로 generalize하여 permutation을 학습하도록 함
      - Invertible하도록 weight의 diagonal 성분만 남김
      - Forward : y = W * x
      - Inverse : $x = W^{-1} * y$ log-det : h * w * log(det|W|)
  - Affine coupling layer
    - RealNVP와 동일한 transformation function의 affine coupling layer 사용

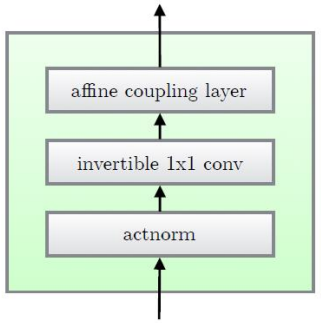


Fig.10 Architecture of a step in Glow

Glow[1]

- Multi-scale architecture
  - 3개의 substep으로 구성된 step of flow를 사용하여 multi-scale architecture 구성
    - Checkerboard pattern을 사용해 이미지를 channel-wise masking하여 squeeze함
    - Step of flow를 K번 수행 하는 것 : Level
    - Level의 수행이 끝나면 transform되는 부분과 그렇지 않은 부분으로 split 함
      - 나뉜 부분 중 transform이 끝난 부분 (z_i)는 바로 Gaussian distribution이 되도록 학습
      - 나머지 절반은 다음 level로 넘겨 다음 level의 step of flow 수행
      - 마지막 level의 경우 연산 결과가 Gaussian distribution이 되도록 학습

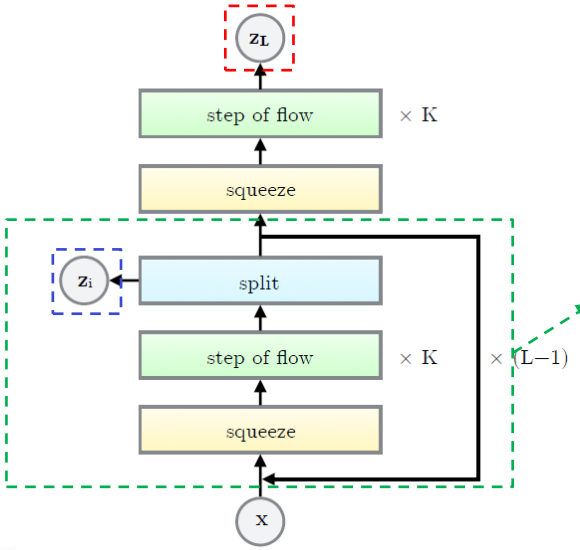


Fig.11 Glow architecture

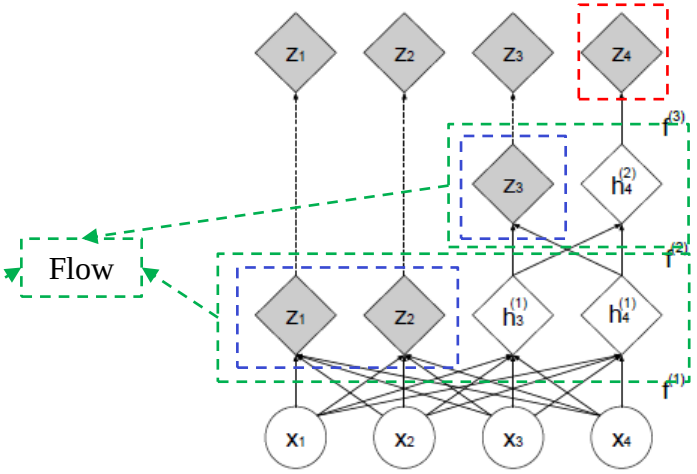


Fig.12 Multi-scale architecture

Glow[1]

- Experimental Results

Fig.13 Random samples from RealNVP

Fig.14 Random samples from Glow

Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	**3.35**	**4.09**	**3.81**	**2.38**	**2.46**	**2.67**

Table.1 Model comparison (in Bits per dimension)

*Bits per dimension (BPD) : Negative Log-Likelihood / (C * H * W)
낮을수록 Gaussian distribution에 근접하도록 잘 학습된 것

Glow[1]

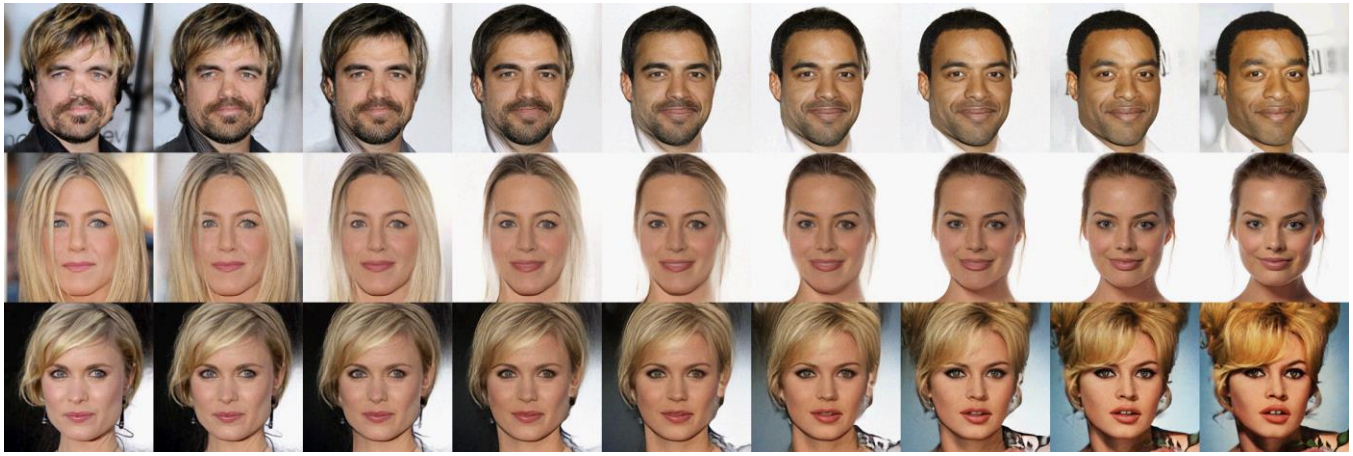


Fig.15 Linear interpolation in latent space between real images

- Ablation study

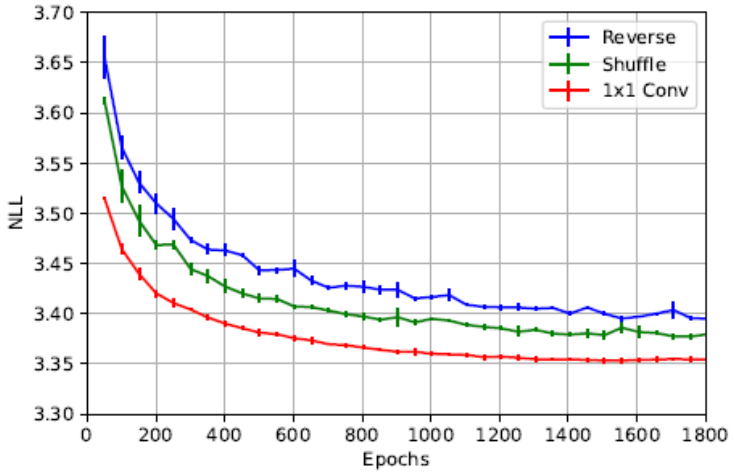


Fig.16 Comparison of the three permutation methods

Conditional Normalizng Flows

- Conditional Normalizing Flows[1]
  - Affine coupling layer에 condition을 부여

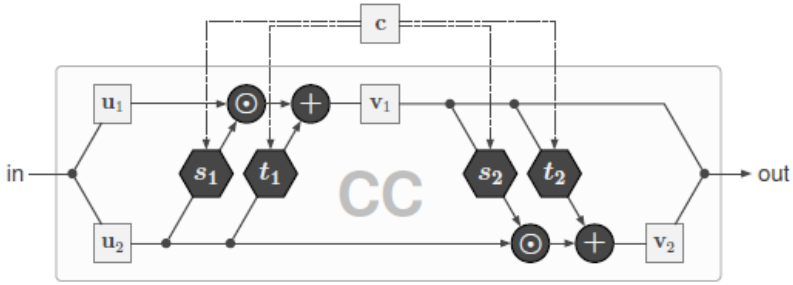


Fig.17 Conditional coupling block (affine transform)

$$f^{-1}(\cdot\,;\mathbf{c},\theta) = g(\cdot\,;\mathbf{c},\theta).$$

$$p_X(\mathbf{x};\mathbf{c},\theta) = p_Z\left(f(\mathbf{x};\mathbf{c},\theta)\right)\left|\det\left(\frac{\partial f}{\partial \mathbf{x}}\right)\right|$$

$$\mathcal{L} = \mathbb{E}_i\left[-\log\left(p_X(\mathbf{x}_i;\mathbf{c}_i,\theta)\right)\right] - \log\left(p_\theta(\theta)\right)$$

Conditional Normalizng Flows

- Conditional Normalizing Flows[1]
  - Class-conditional generation for MNIST dataset
    - One-hot vector condition으로 class condition을 주어 학습
    - Tidy, narrow, wide, messy, faint, bold의 총 6가지 종류의 dataset 학습
      - 0 ~ 9의 숫자 condition, 글자 종류의 두 가지 condition을 학습하였음
        - ✓ 0~9에 해당하는 이미지가 condition을 통해 하나의 latent space로 학습됨
      - Backward propagation에서는 latent space에서 normal distribution에 따라 sample하고 생성하고자 하는 숫자의 condition을 사용하여 inverse 방향 inference

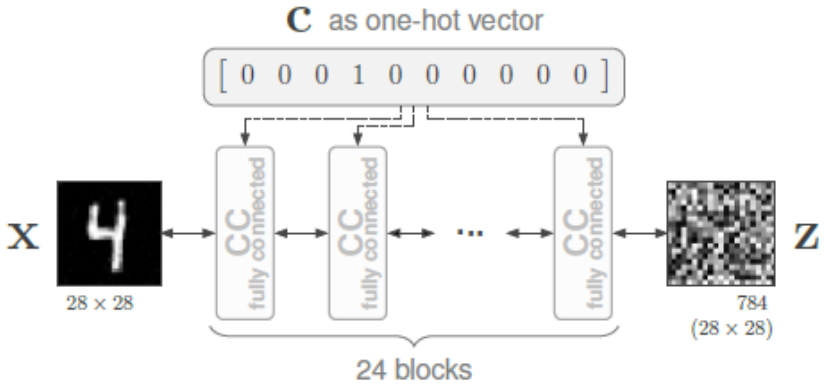


Fig.18 Normalizing Flows model for MNIST generation

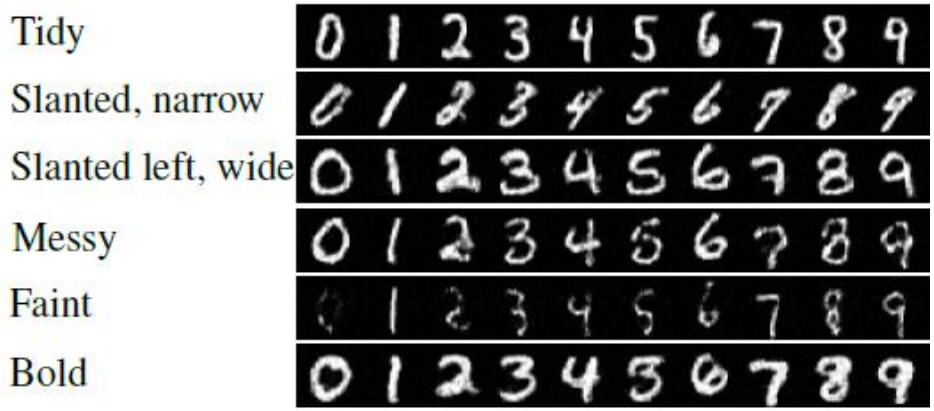


Fig.19 Generated samples from conditional flow model

Conditional Normalizng Flows

- Conditional Normalizing Flows[1]
  - Diverse ImageNet colorization
    - YCbCr image의 Y를 condition으로 사용하여 CbCr 성분을 변형하여 다양한 색상의 이미지 생성
    - Y로부터 color feature CbCr을 예측하도록 condition feature extraction network h를 pretrain하여 사용함
      - 연산량 최소화를 위해 각 transform function에서 사용할 수 있는 condition network의 크기는 제한됨
        - ✓ Network h를 사용해 각 layer에서는 network h에서 생성한 feature를 각 flow level에 맞게 tuning하는 역할을 수행

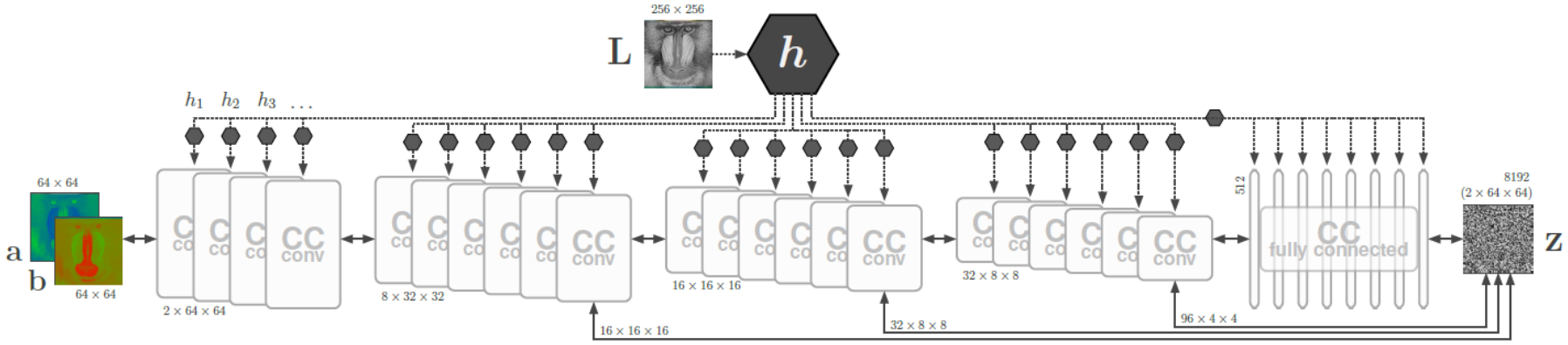
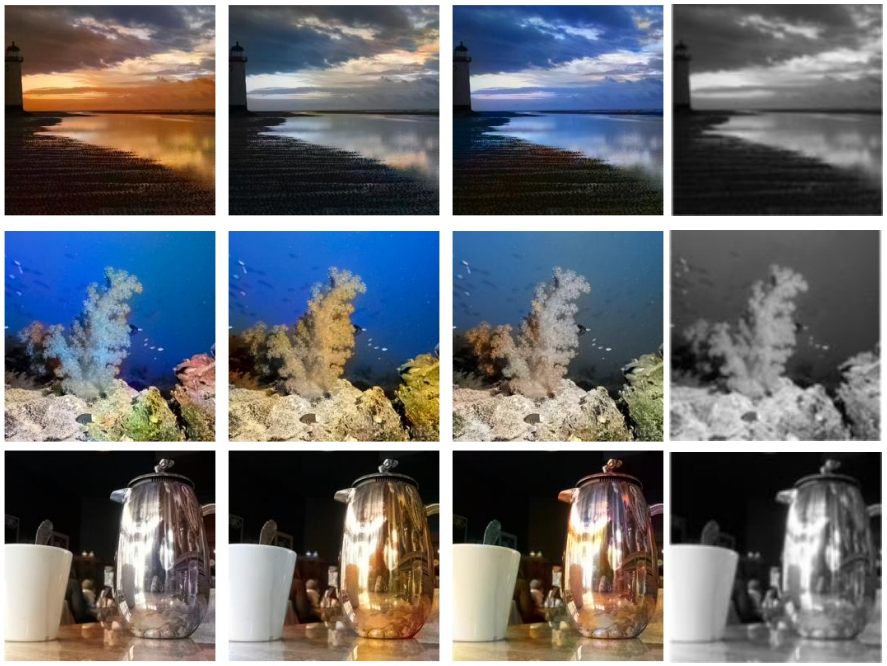


Fig.20 Normalizing Flows model for ImageNet colorization

Conditional Normalizng Flows

- Conditional Normalizing Flows[1]
  - Diverse ImageNet colorization



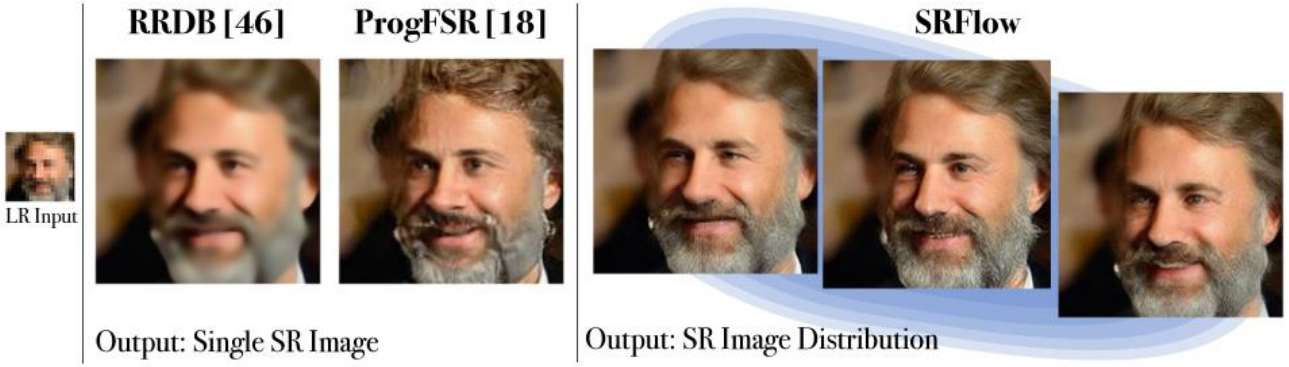
<Generated image> <Condition>

<Generated image>

Fig.21 Samples from Conditional Normalizing Flows

SRFlow[1] [ECCV 2020 Spotlight]

- Super Resolution
  - Low resolution(LR)의 이미지로부터 high resolution(HR) 이미지를 복원
    - 한 장의 LR 이미지로부터 여러 HR이미지가 생성될 수 있음
      - 여러 장의 다른 HR 이미지를 resize하였을 때 동일한 LR 이미지를 얻을 수 있기 때문
        - ✓ ill-posed problem
    - 기존의 GAN 방식으로 SR을 해결하려고 한 문제는 이를 반영하지 못함
      - Deterministic한 방식으로, 하나의 weight에서 하나의 output만 생성이 가능함
    - 따라서 Normalizing Flows를 도입하여 이러한 특성을 고려할 수 있는 method 제안
      - Content loss, adversarial loss로 구성된 GAN
      - 본 논문에서는 Normalizing Flow를 사용하여 negative log-likelihood loss만 사용



SRFlow[1]

- Main contribution
  - ill-posed problem인 super resolution의 nature를 반영 가능
  - 단일 loss (NLL)을 사용하여 mode-collapse가 일어나는 GAN보다 학습이 안정적
    - Mode-collapse : Discriminator가 Generator보다 쉽게 성능이 좋아져 generator가 하나의 sample만 생성하게 되는 것
  - 정확한 latent space로의 mapping이 보장됨
    - Latent space에서의 manipulation technique을 통한 image content transfer, latent space normalization을 통한 harmonization이 적용될 수 있음

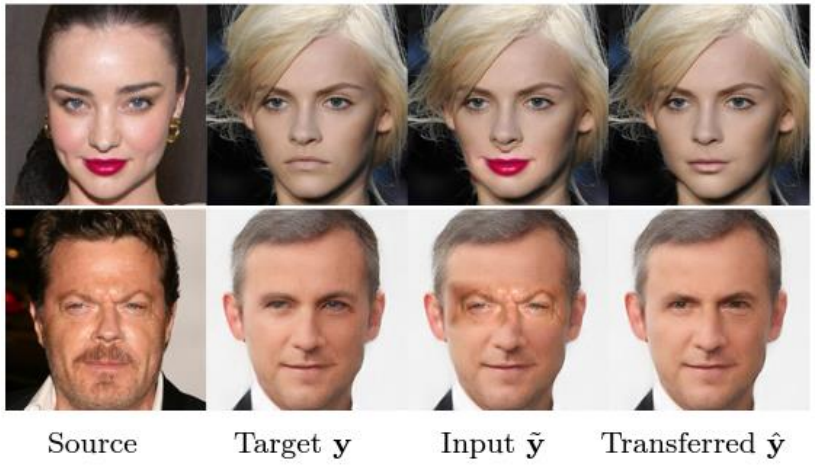


Fig.22 Sample images of content transfer

SRFlow[1]

- Conditional Affine Coupling
  - Affine coupling layer에 condition을 적용하여 HR 이미지에 대한 분포 학습
    - LR 이미지를 condition으로 주어 HR 이미지의 분포를 학습
    - Affine layer의 neural network input으로 split한 channel의 절반과 LR image condition을 concatenate하여 입력
      - Neural network의 결과로 나온 scale factor, translate factor를 사용하여 나머지 절반 channel의 affine transform을 수행

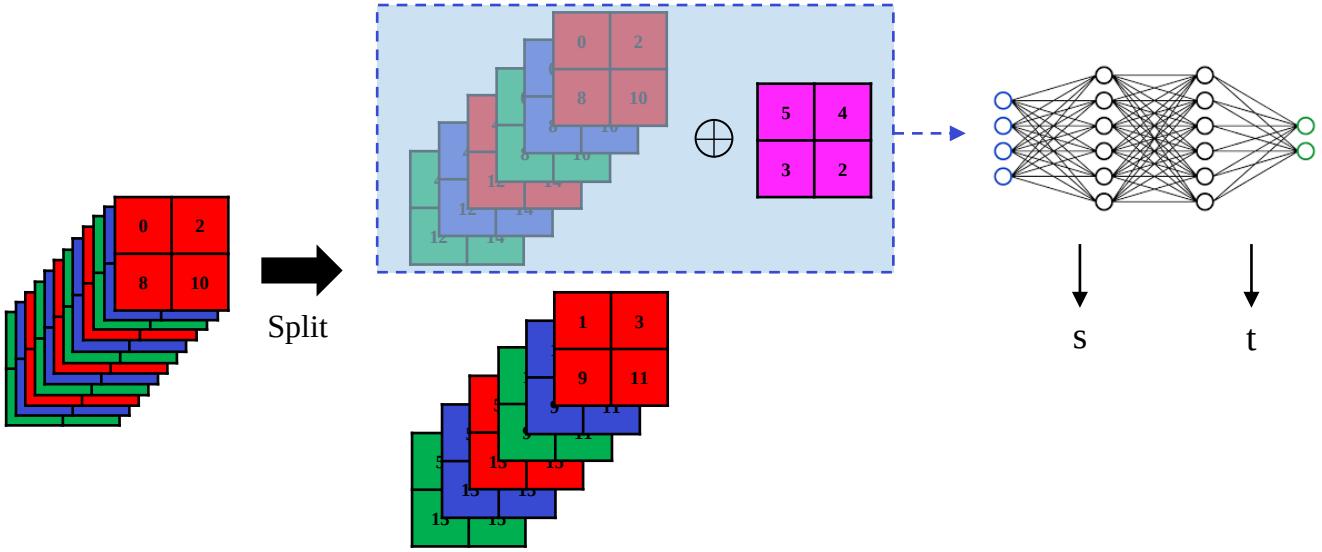


Fig.23 Conditional affine coupling block structure

SRFlow[1]

- GLOW architecture를 따름
  - 1x1 convolution
    - 최적의 channel 순서를 학습
  - Actnorm
    - Invertible한 channel-wise normalization
  - Squeeze
    - RealNVP와 동일한 squeeze operation
  - Flow-step
    - 각 flow-step은 Actnorm, 1x1 conv, affine injector layer, conditional affine coupling layer로 구성
    - 각 level마다 16개의 steps로 구성
    - 4x SR의 경우 3개의 level으로 구성
    - 8x SR의 경우 4개의 level으로 구성

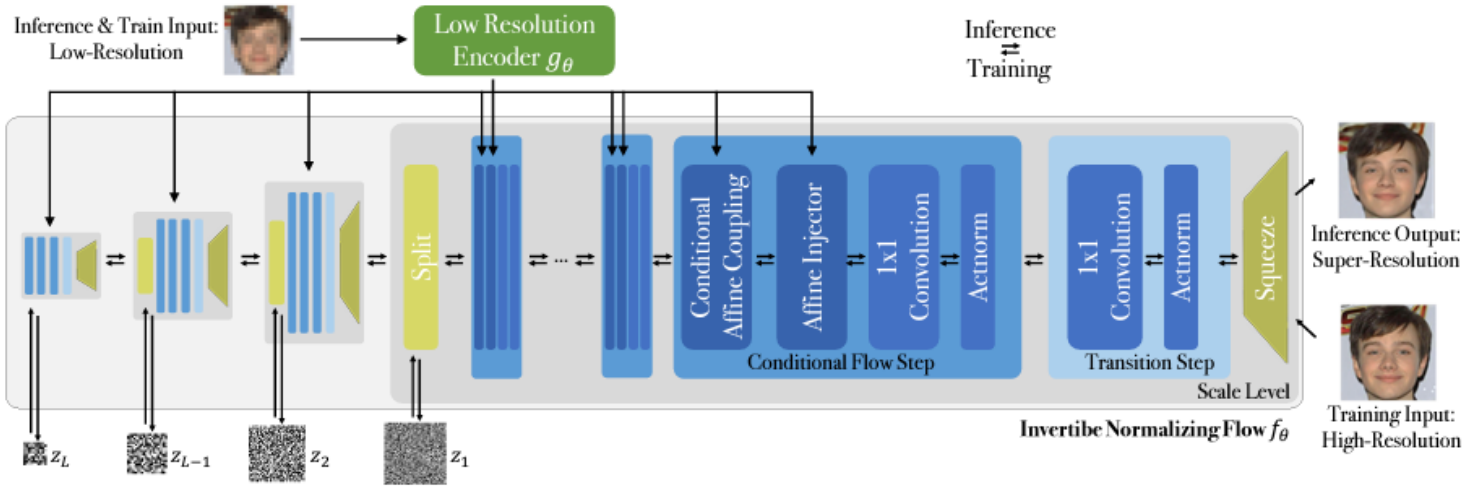


Fig.24 SRFlow architecture

1) Lugmayr, Andreas, et al. "Srflow: Learning the super-resolution space with normalizing flow." *European conference on computer vision*. Springer, Cham, 2020.
2) Wang, Xintao, et al. "Esrgan: Enhanced super-resolution generative adversarial networks." *Proceedings of the European conference on computer vision (ECCV) workshops*. 2018.

SRFlow[1]

- Affine injector layer
  - Condition만을 이용하여 affine coupling layer 수행
    - 따라서 channel을 split하여 neural network에 넣지 않음
    - 오직 condition으로 생성한 scale, transition 값을 전체 feature에 대하여 affine 수행
    - 이 layer를 사용해 condition에 따른 분포 학습을 도움
- Low resolution encoder
  - 각 level에서의 LR condition의 풍부한 표현력을 위해 ESRGAN[2]에서 제안한 RRDB 구조를 사용하여 encoding하여 사용
- Transition step
  - Squeezing operation에 의해 나타나는 checkerboard artifact를 완화
    - Squeezing operation은 단순히 pixel의 re-ordering이므로 artifact가 발생할 수 있음
    - 기존 방법은 모든 step을 condition layer로 구성하였는데, 이 경우 squeezing operation에 대한 channel permutation을 학습하지 못하는 경우가 발생함
    - Squeezing 이후에 conditional layer 없이 1x1 conv만 통과함으로써 squeeze 이후 channel 순서를 더 잘 학습할 수 있음

DIV2K 4×	PSNR↑	SSIM↑
No Lin. F-Step	26.96	0.759
No Affine Inj.	26.81	0.756
SRFlow	27.09	0.763

Table.2 Comparison with ablative experiments

SRFlow[1]

- Train
  - High resolution 이미지를 squeeze하여 input으로 사용
  - Low resolution 이미지를 LR encoder를 통해 condition으로 사용
    - 결과적으로 latent space에서 Gaussian distribution을 갖도록 NLL Loss로 학습
- Inference
  - Latent space에서 Gaussian distribution에서 sample
  - LR 이미지를 condition으로 사용하여 역방향으로 inference하여 SR 이미지 획득

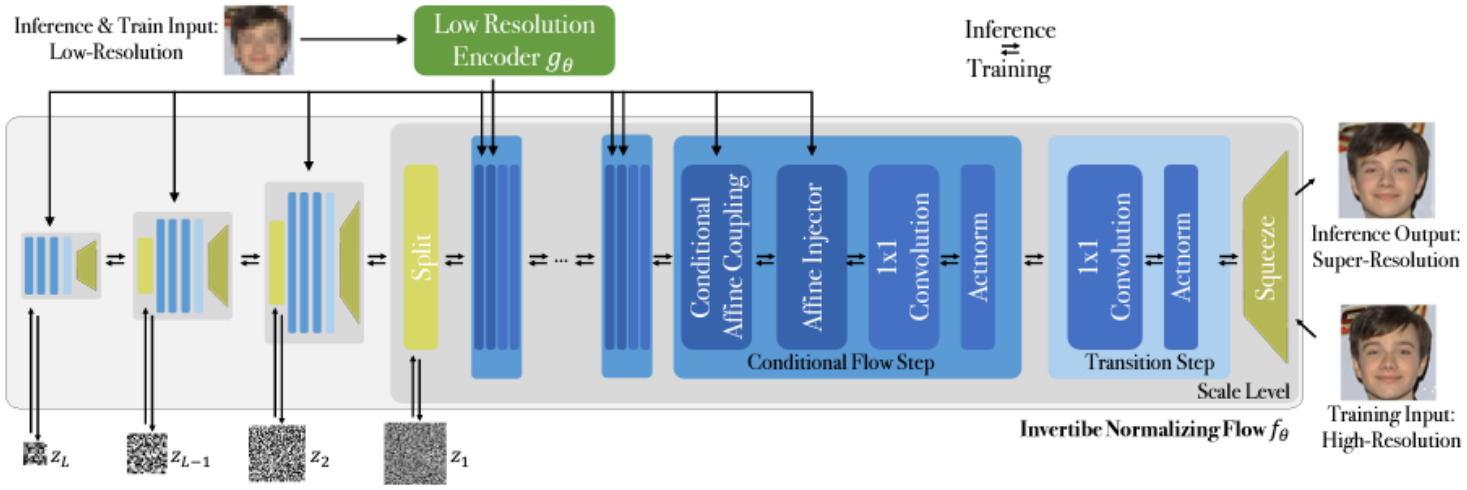


Fig.24 SRFlow architecture

SRFlow[1]

- Experimental Results

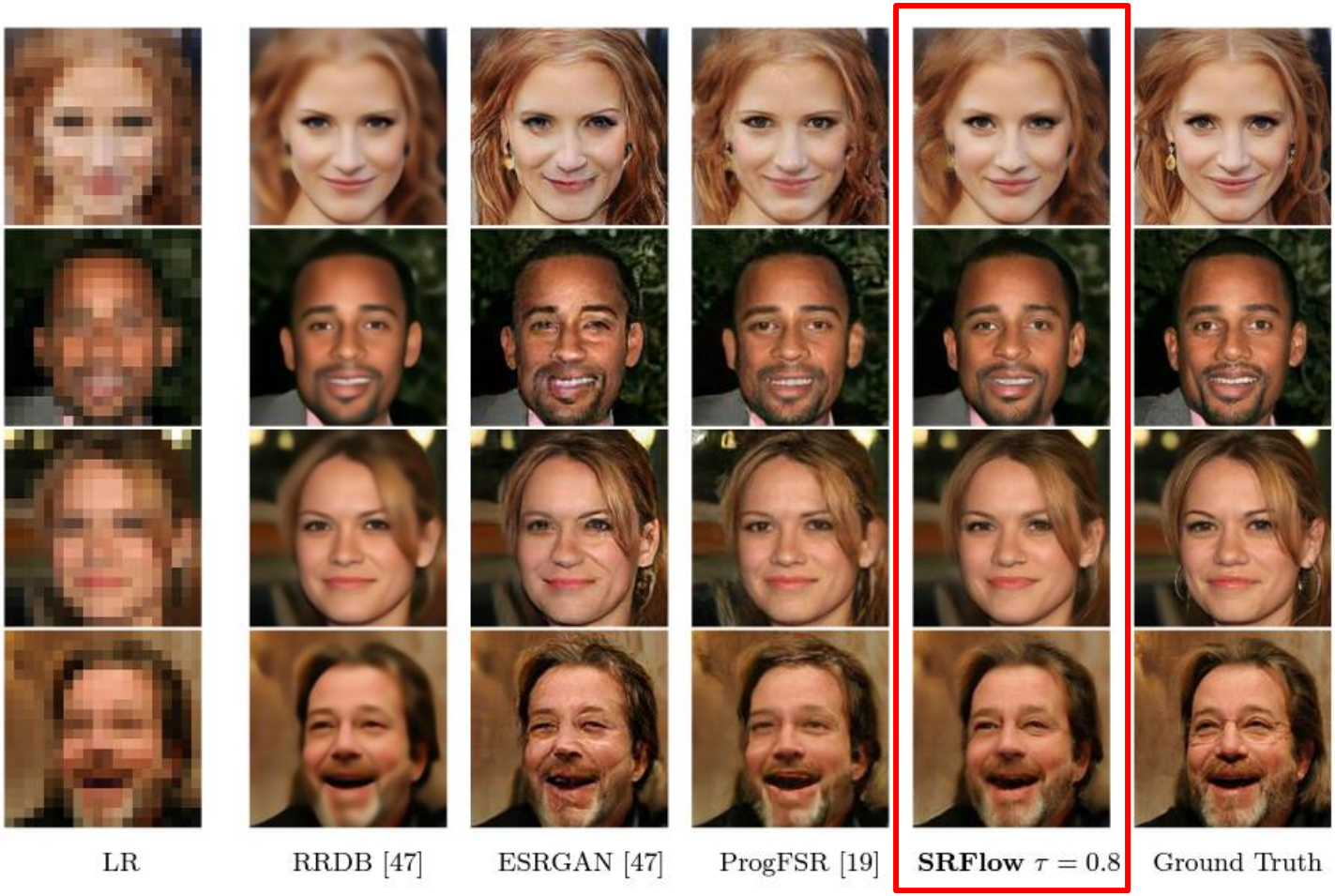


Fig.25 Comparison of our SRFlow with state-of-the-art for face SR on CelebA (x8)

SRFlow[1]

- Experimental Results
  - DIV2K dataset의 100 validation images로 실험

	DIV2K 4×							DIV2K 8×						
	PSNR↑	SSIM↑	LPIPS↓	LR-PSNR↑	NIQE↓	BRISQUE↓	PIQUE↓	PSNR↑	SSIM↑	LPIPS↓	LR-PSNR↑	NIQE↓	BRISQUE↓	PIQUE↓
Bicubic	26.70	0.77	0.409	38.70	5.20	53.8	86.6	23.74	0.63	0.584	37.16	6.65	60.3	97.6
EDSR [24]	28.98	0.83	0.270	54.89	4.46	43.3	77.5	-	-	-	-	-	-	-
RRDB [47]	29.44	0.84	0.253	49.20	5.08	52.4	86.7	25.50	0.70	0.419	45.43	4.35	42.4	79.1
RankSRGAN [56]	26.55	0.75	0.128	42.33	2.45	17.2	20.1	-	-	-	-	-	-	-
ESRGAN [47]	26.22	0.75	0.124	39.03	2.61	22.7	26.2	22.18	0.58	0.277	31.35	2.52	20.6	25.8
SRFlow $\tau = 0.9$	27.09	0.76	0.120	49.96	3.57	17.8	18.6	23.05	0.57	0.272	50.00	3.49	20.9	17.1

Table.3 Result comparison on DIV2K dataset

- CelebA dataset

LR		↑PSNR	↑SSIM	↓LPIPS	↑LR-PSNR	↓NIQE	↓BRISQUE	↓PIQUE	↑Diversity σ
Bicubic	Bicubic	23.15	0.63	0.517	35.19	7.82	58.6	99.97	0
	RRDB [47]	26.59	0.77	0.230	48.22	6.02	49.7	86.5	0
	ESRGAN [47]	22.88	0.63	0.120	34.04	3.46	23.7	32.4	0
	SRFlow $\tau = 0.8$	25.24	0.71	0.110	50.85	4.20	23.2	24.0	5.21
Prog.	ProgFSR [19]	23.97	0.67	0.129	41.95	3.49	28.6	33.2	0
	SRFlow $\tau = 0.8$	25.20	0.71	0.110	51.05	4.20	22.5	23.1	5.28

Table.4 Result comparison on CelebA dataset

SRFlow[1]

- Ablative Study
  - 각 level을 구성하는 step 개수
    - Step이 많을수록 complex한 구조에서의 artifact가 적고 reconstruction 결과가 더 자연스럽게 나타남
  - Coupling layer를 구성하는 neural network의 channel dimension
    - Dimension이 클수록 자연스러운 이미지 생성 가능

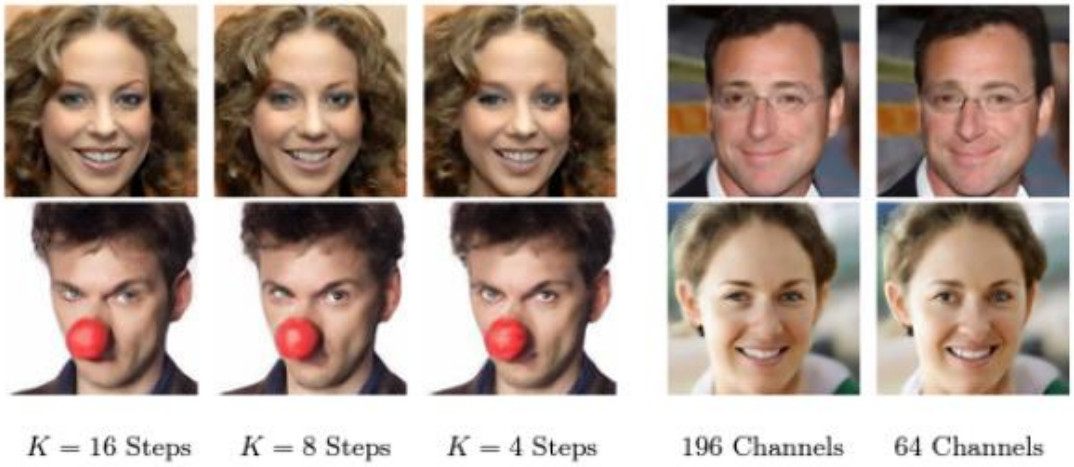


Fig.26 Analysis of number of flow steps and dimensionality

감사합니다