DL Seminar

SRGAN

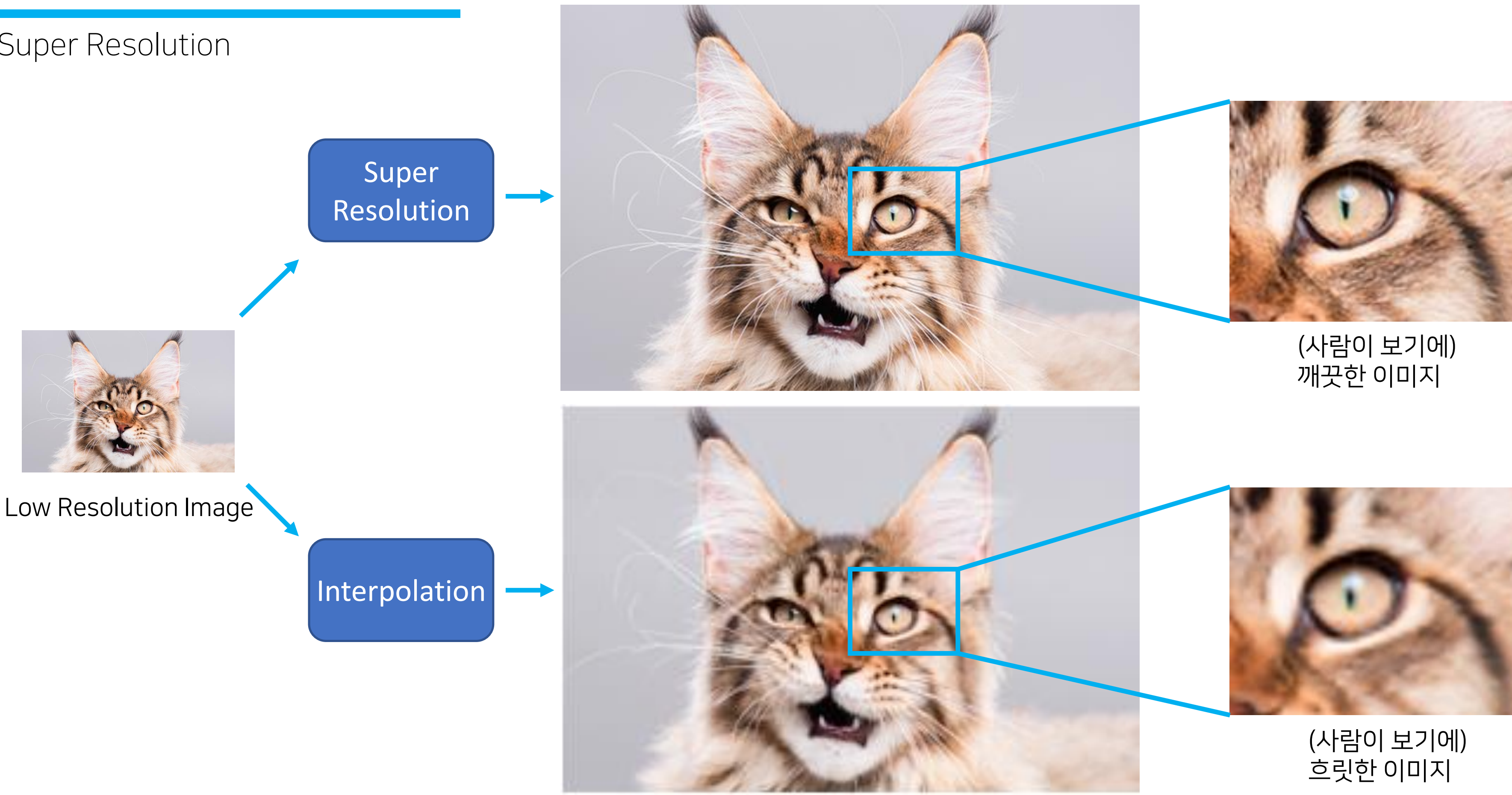
Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

한양대학교
HANYANG UNIVERSITY

인공지능 연구실
김지성

Introduction

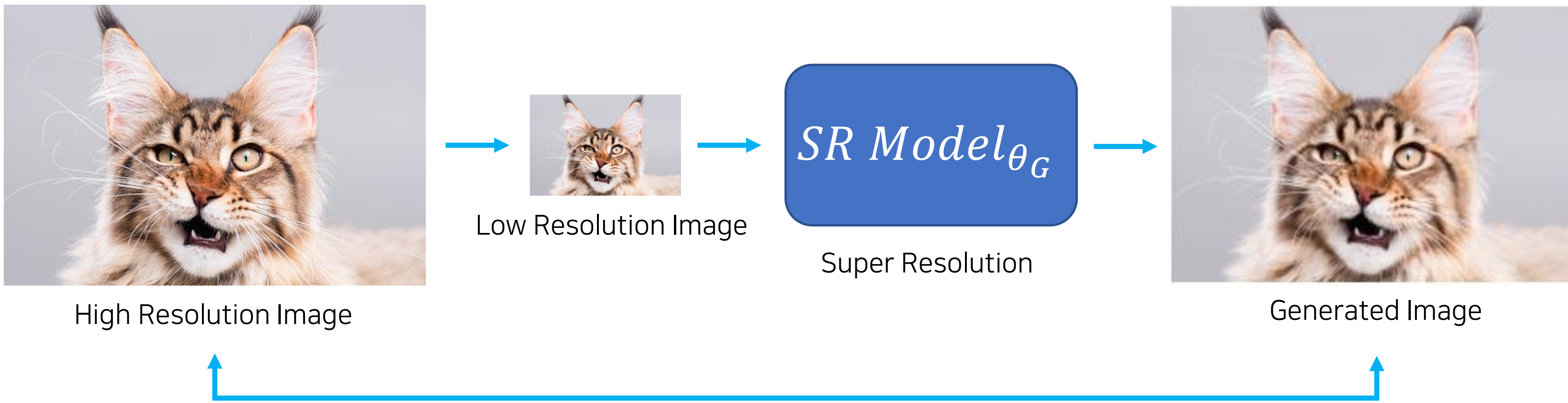
Super Resolution



High Resolution Image

Introduction

Super Resolution



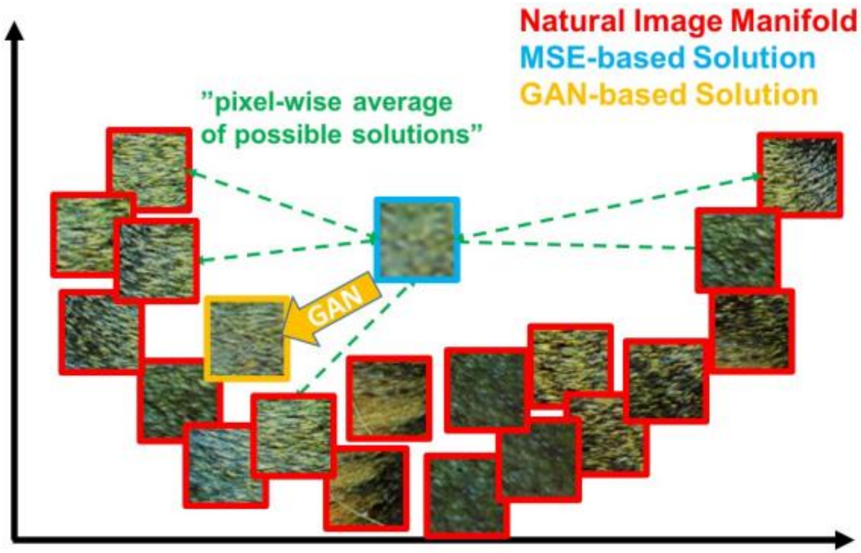
차이를 최소화 하는 θ_G 찾기

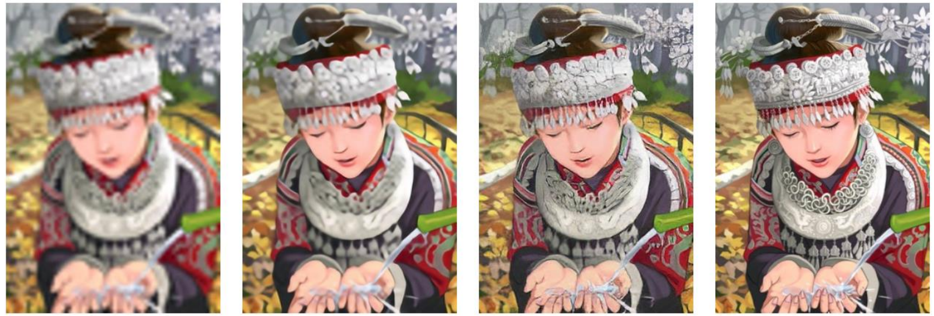
$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{n=1}^{N} l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR}) \qquad (1)$$

I^{HR} : 저해상도 이미지
I^{LR} : 저해상도 이미지
l^{SR} : Super Resolution Loss Function
θ_G : SR 모델의 Weight, bias

Method

Adversarial loss





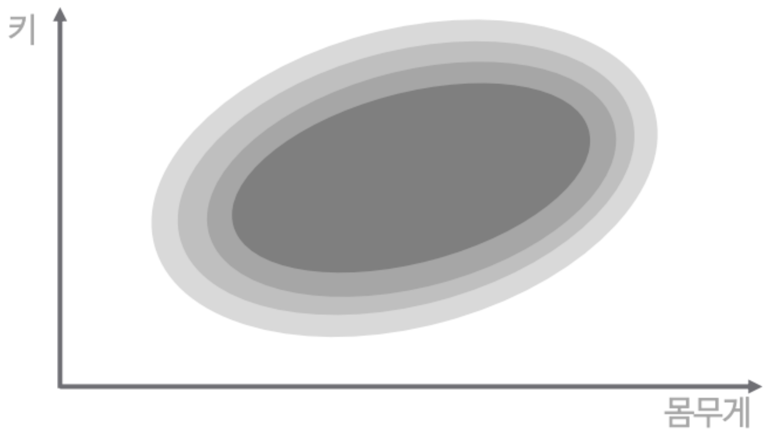
bicubic (21.59dB/0.6423) | SRResNet (23.53dB/0.7832) | SRGAN (21.15dB/0.6868) | original

SRResnet

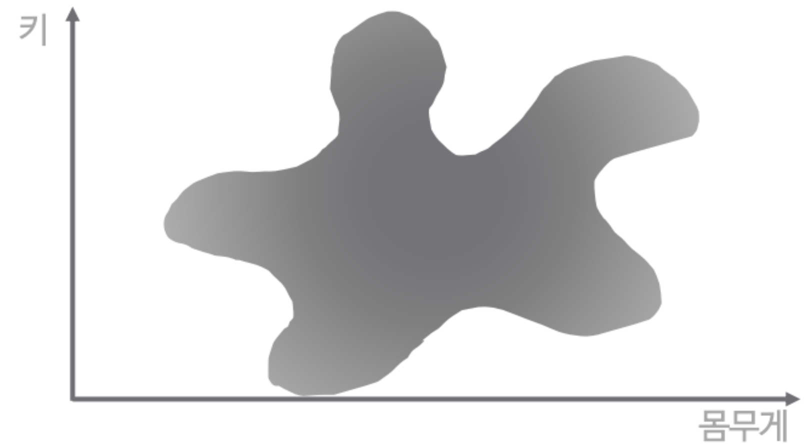
SRGAN

Method

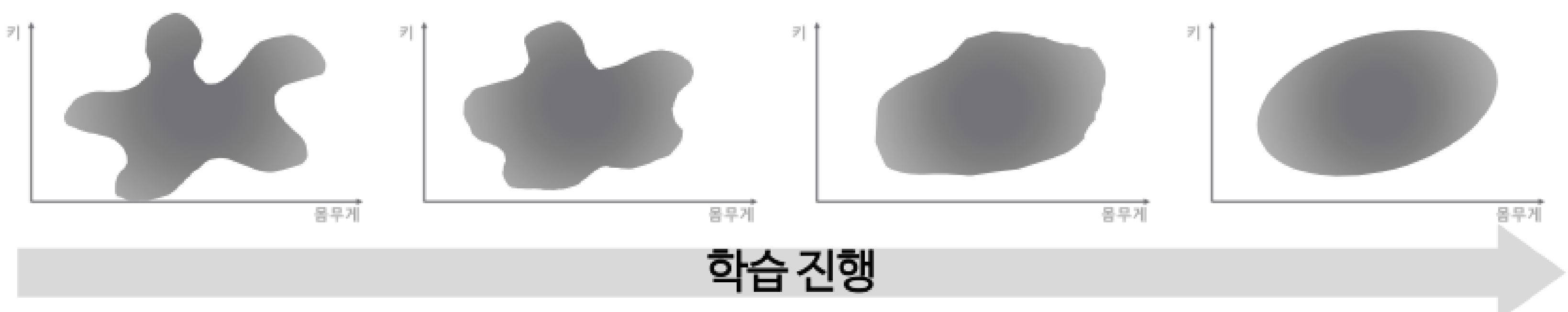
GAN



Target Data



Generated Data



Method

GAN process

$$\min_{\theta_G}\max_{\theta_D} \mathbb{E}_{I^{HR}\sim p_{\text{train}}(I^{HR})}[\log D_{\theta_D}(I^{HR})]+ \mathbb{E}_{I^{LR}\sim p_G(I^{LR})}[\log(1-D_{\theta_D}(G_{\theta_G}(I^{LR}))] \tag{2}$$

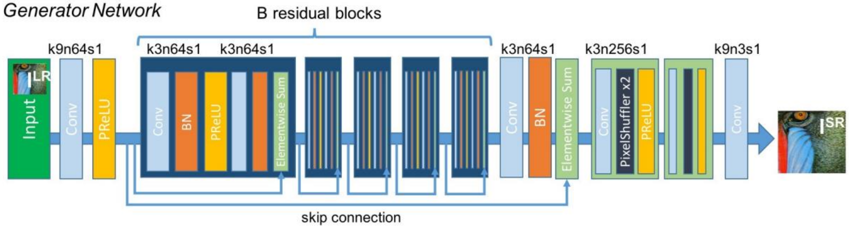
I^{HR} : 저해상도 이미지
I^{LR} : 저해상도 이미지
D_{θ_D} : Discriminator
G_{θ_G} : Generator

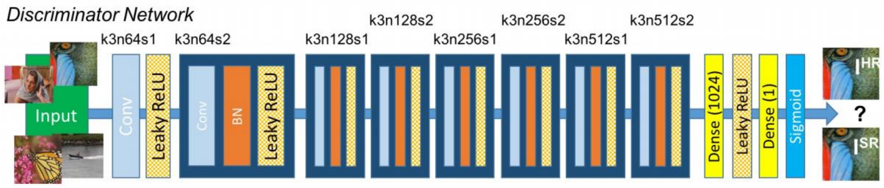
Pseudocode – Adversarial Training

```
lr_image = tf.placeholder('float32', [batch_size, 96, 96, 3]) #저해상도 이미지
hr_image = tf.placeholder('float32', [batch_size, 384, 384, 3]) #고해상도 이미지

logits_real = Discriminator(hr_image)
logits_fake = Discriminator(Generator(lr_image))

d_loss = tl.cost.sigmoid_cross_entropy(logits_real, tf.ones_like(logits_real))
d_loss += tl.cost.sigmoid_cross_entropy(logits_fake, tf.zeros_like(logits_fake))
g_loss = tl.cost.sigmoid_cross_entropy(logits_fake, tf.ones_like(logits_fake))

g_optim = tf.train.AdamOptimizer(lerning_rate=1e-4, beta1=0.9).minimize(g_loss)
d_optim = tf.train.AdamOptimizer(lerning_rate=1e-4, beta1=0.9).minimize(d_loss)

for epoch in range(0, n_epoch + 1):
    sess.run(d_optim, {low_image: low_imgs_96, target_image: imgs_384})
    sess.run(g_optim, {low_image: low_imgs_96, target_image: imgs_384})
```

Method

G,D model





Pseudocode - Residual blocks

```
for i in range(16):
    nn = Conv2d(n, 64, (3, 3), (1, 1), act=None, padding='SAME')
    nn = BatchNormLayer(nn, act=tf.nn.relu)
    nn = Conv2d(nn, 64, (3, 3), (1, 1), act=None, padding='SAME')
    nn = BatchNormLayer(nn)
    nn = ElementwiseLayer([n, nn], tf.add)
    n = nn
```

Pseudocode - UpSampling

```
n = UpSampling2dLayer(n, size=[width * 2, height * 2], method=NEAREST_NEIGHBOR)
n = Conv2d(n, 64, (3, 3), (1, 1), padding='SAME')
n = BatchNormLayer(n, act=tf.nn.relu)
```

Method

Content loss

$$l^{SR} = \underbrace{\underbrace{l^{SR}_{X}}_{\text{content loss}} + \underbrace{10^{-3} l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}} \quad (3)$$

l^{SR}_{X} : Content loss, 픽셀간 유사성 대신 지각적 유사성 제공
l^{SR}_{Gen} : SR이미지를 자연스러운 이미지 매니폴드로 유도

Artifacts in GAN generated image

Method

Content loss 1

$$l^{SR}_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I^{HR}_{x,y} - G_{\theta_G}(I^{LR})_{x,y})^2 \qquad (4)$$

r : 다운샘플링 계수
W, H : LR 이미지의 Width, Height
G : SR모델
θ_G : SR 모델의 Weight, bias
x, y : 픽셀 x, y

Pseudocode – Content loss(MSE)

```
mse_loss = tl.cost.mean_squared_error(Generator(lr_image), hr_image)
```

Method

Content loss 2

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \quad (5)$$

$\phi_{i,j}$: VGG19 net에서 i번째 맥스풀링 전, j번째 컨볼루션 레이어에 의해 얻어진 피쳐맵
$W_{i,j}$, $H_{i,j}$: 피쳐맵의 차원

Pseudocode – Content loss(VGGnet)

```
vgg_loss = tl.cost.mean_squared_error(vggNet(Generator(lr_image)), vggNet(hr_image))
```

Method

SR loss

$$l^{SR} = \underbrace{\underbrace{l^{SR}_{X}}_{\text{content loss}} + \underbrace{10^{-3} l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}} \quad (3)$$

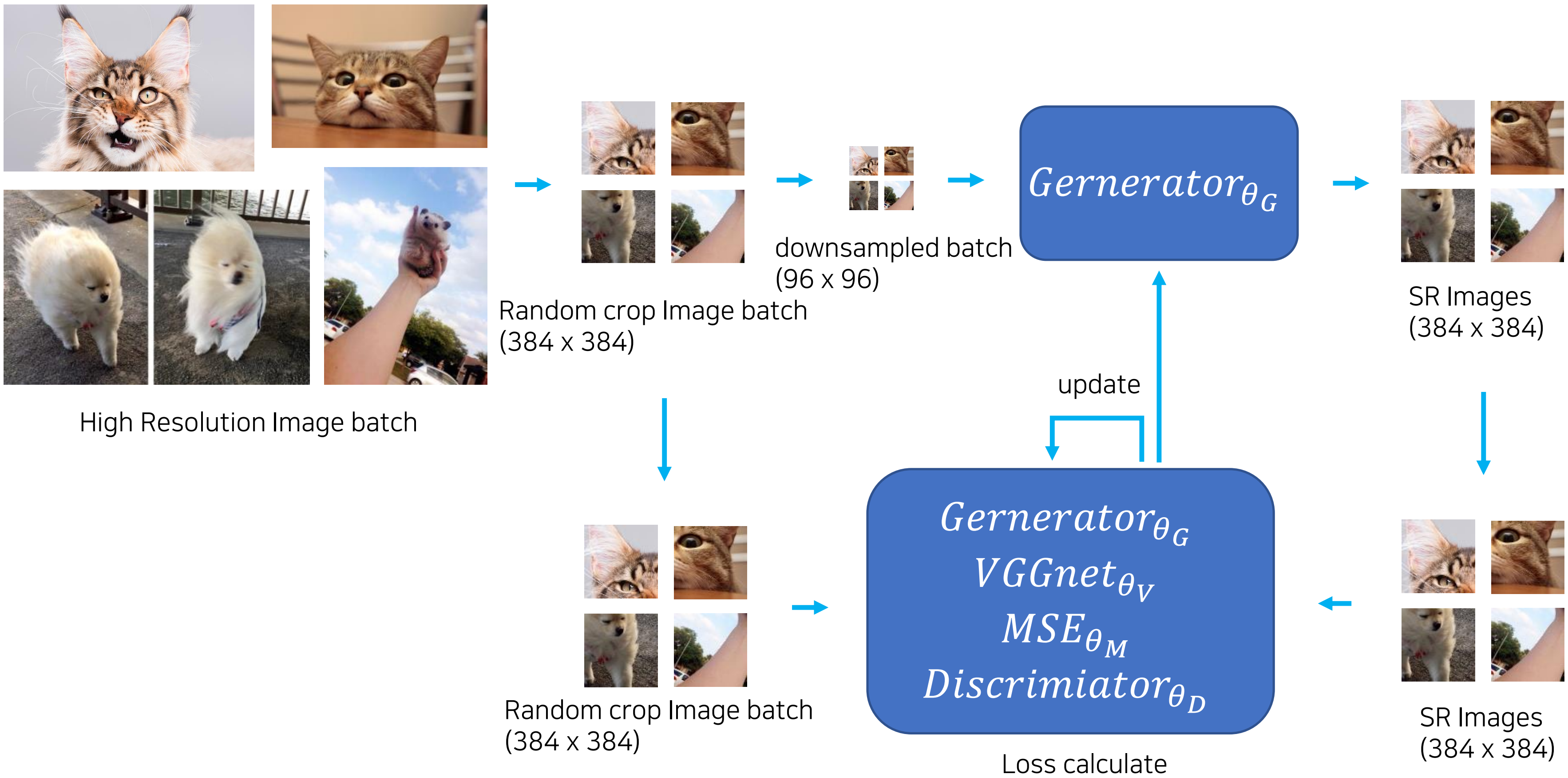
l^{SR}_{X} : Content loss, 픽셀간 유사성 대신 지각적 유사성 제공
l^{SR}_{Gen} : SR이미지를 자연스러운 이미지 매니폴드로 유도

Pseudocode – final SR loss

```
d_loss = tl.cost.sigmoid_cross_entropy(logits_real, tf.ones_like(logits_real))
d_loss += tl.cost.sigmoid_cross_entropy(logits_fake, tf.zeros_like(logits_fake))

g_loss = tl.cost.sigmoid_cross_entropy(logits_fake, tf.ones_like(logits_fake))
mse_loss = tl.cost.mean_squared_error(Generator(lr_image), hr_image)
vgg_loss = tl.cost.mean_squared_error(vggNet(Generator(lr_image)), vggNet(hr_image))
g_loss = g_loss + mse_loss + vgg_loss
```

Experiments

Training Process



Experiments

Benchmark

학습 파라미터
Random crop Size $= 386$
Downsampling factor $r = 4$
G loss 가중치 = $1e - 3$
MSE loss 가중치 = 1
VGG loss 가중치 = $2e - 6$

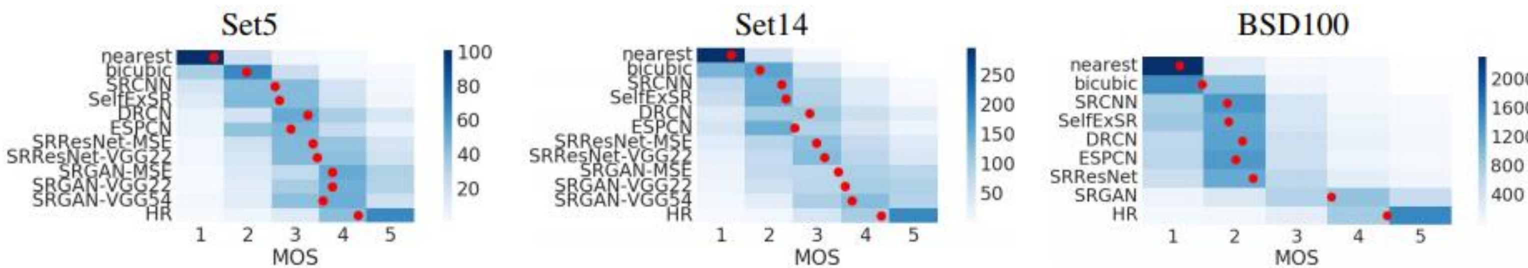
학습 데이터셋
ImageNet 35만개 이미지

Experiments

Benchmark

	SRResNet-		SRGAN-		
Set5	MSE	VGG22	MSE	VGG22	VGG54
PSNR	32.05	30.51	30.64	29.84	29.40
SSIM	0.9019	0.8803	0.8701	0.8468	0.8472
MOS	3.37	3.46	3.77	3.78	3.58
Set14					
PSNR	28.49	27.19	26.92	26.44	26.02
SSIM	0.8184	0.7807	0.7611	0.7518	0.7397
MOS	2.98	3.15*	3.43	3.57	3.72*

Set5	nearest	bicubic	SRCNN	SelfExSR	DRCN	ESPCN	**SRResNet**	**SRGAN**	HR
PSNR	26.26	28.43	30.07	30.33	31.52	30.76	**32.05**	29.40	∞
SSIM	0.7552	0.8211	0.8627	0.872	0.8938	0.8784	**0.9019**	0.8472	1
MOS	1.28	1.97	2.57	2.65	3.26	2.89	3.37	**3.58**	4.32
Set14									
PSNR	24.64	25.99	27.18	27.45	28.02	27.66	**28.49**	26.02	∞
SSIM	0.7100	0.7486	0.7861	0.7972	0.8074	0.8004	**0.8184**	0.7397	1
MOS	1.20	1.80	2.26	2.34	2.84	2.52	2.98	**3.72**	4.32
BSD100									
PSNR	25.02	25.94	26.68	26.83	27.21	27.02	**27.58**	25.16	∞
SSIM	0.6606	0.6935	0.7291	0.7387	0.7493	0.7442	**0.7620**	0.6688	1
MOS	1.11	1.47	1.87	1.89	2.12	2.01	2.29	**3.56**	4.46



PSNR = $10log\frac{255^2}{MSE}$

최대 신호 대 잡음비

단위는 db 이며, 손실이 적을수록 높은 값을 가짐

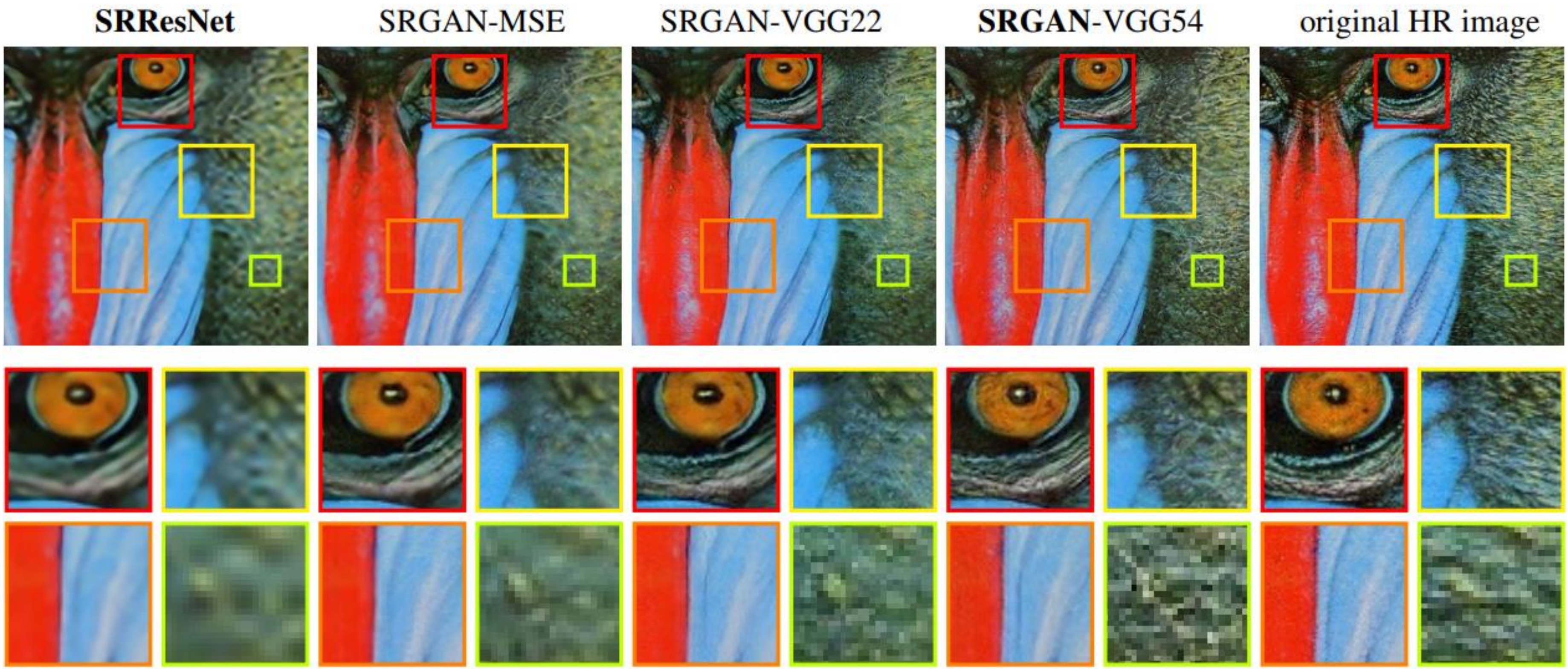
Mean Opinian Score

평가자 : 26명

점수 : 1(나쁜 품질) ~ 5(좋은 품질) 점

Experiments

Benchmark



Differences in results according to Content loss

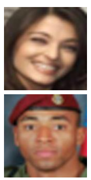
Experiments

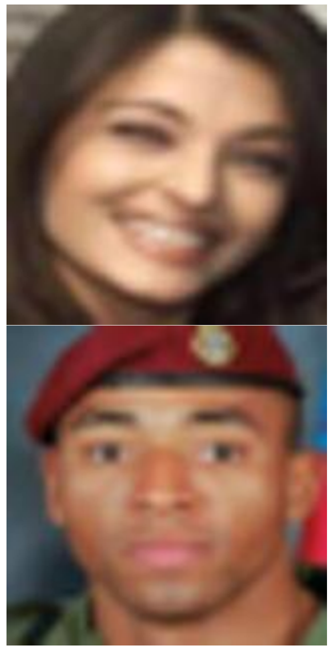
Facenet Benchmark

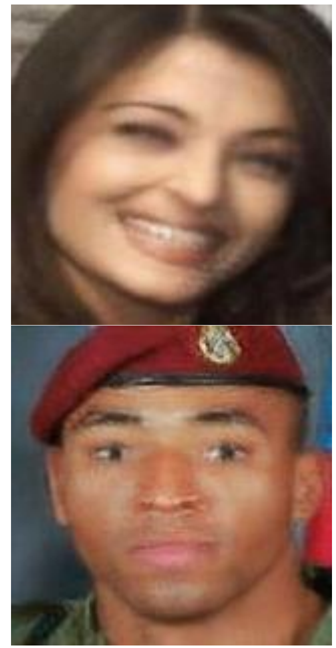
96 x 96 Low Resolution Image

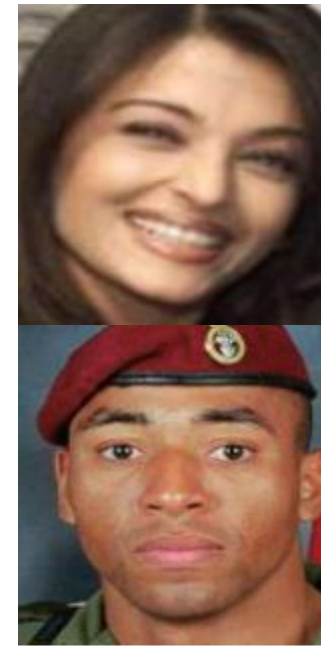
386 x 386 Super Resolution Image

Experiments


40 x 40 LR

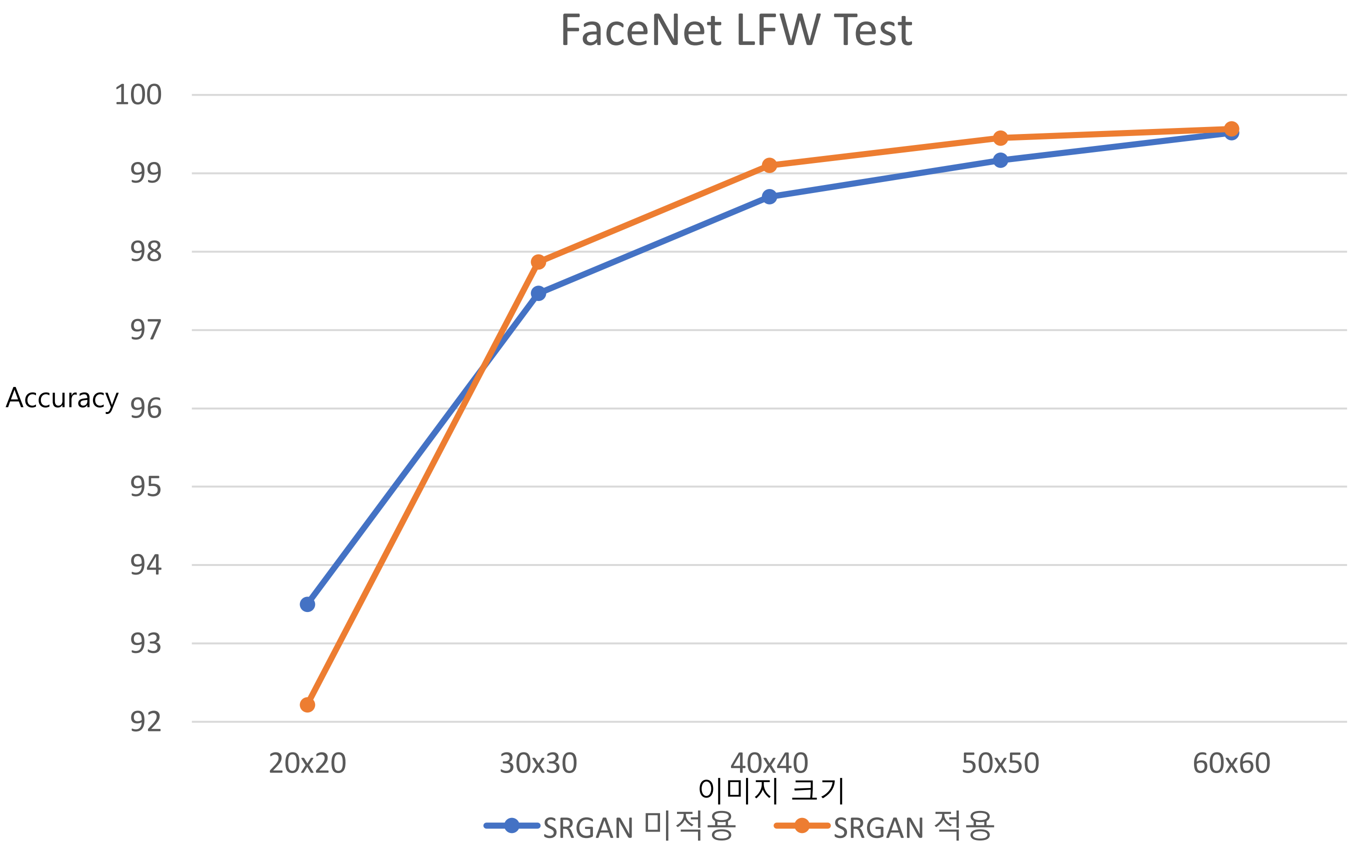

160 x 160 bicubic


160 x 160 SR


160 x 160 HR

Experiments

Facenet Benchmark



	SRGAN 미적용	SRGAN 적용
20x20	93.5	92.217
30x30	97.467	97.867
40x40	98.7	99.1
50x50	99.167	99.45
60x60	99.517	99.567

감사합니다.