Paper Review 10 - SRNTT [Super Resolution]

Image Super-Resolution by Neural Texture Transfer

- Accepted Conference Name & Year : CVPR 2019
- 1st Author Name & Institute : Zhifei Zhang, Adobe

Contribution

- Super Resolution 에서는 Sing Image Super Resolution 이 제일 흔하다. 즉, 단일 이미지 한장을 주면 그 안에서 최대한 고해상도 이미지를 복원하는 것이다. 딥러닝으로 넘어오고나서도 SISR 모델들은 굉장히 많이 등장했지만, 결국 단일 이미지만 가지고 고해상도로 복원해야된다는 부분에서 생기는 정보 부족 문제는 해결하지 못하고 있다.
- 물론 Hand-Craft 한 SR 방식 중에는 Reference 이미지를 제공해 좀 더 사용할 정보를 늘리는 RefSR 방식들도 있다. 이 방식은 간단히 말하자면 SR 할 이미지와 유사한 Reference 이미지의 고해상도 정보를 텍스쳐 정보로 사용하는 것이다. 다만, 유사한 Reference 를 일일이 찾는 것도 쉽지 않고 만약 유사하지 않은 이미지가 제공되면 SISR보다도 형편없는 결과를 초래하게 된다는 단점이 존재한다.
- 그래도 SISR 만으로는 한계가 있으니 RefSR의 개발도 꾸준히 진행되고 있는데, 딥러닝 모델로는 대표적으로 CrossNet 이 있다고 한다.
- 하지만 딥러닝 모델에서도 유사하지 않는 Reference Image가 제공되면 좋지 않는 성능이 나오는 것은 여전한데, 이번 논문에서 제안하는 SRNTT는 그런 경우가 발생해도 최소한 SISR 모델의 성능을 보장한다고 주장한다.
- 또, Style Transfer처럼 Reference Image의 feature 를 뽑아 그것을 Texture 로써 사용해서 Reference Image의 고해상도 정보도 충분히 잘 활용하도록 구성했다. Style Transfer 등에 자주 사용되는 VGG19를 Texture Transfer 모델로 사용했다고 한다.

Proposed Architecture

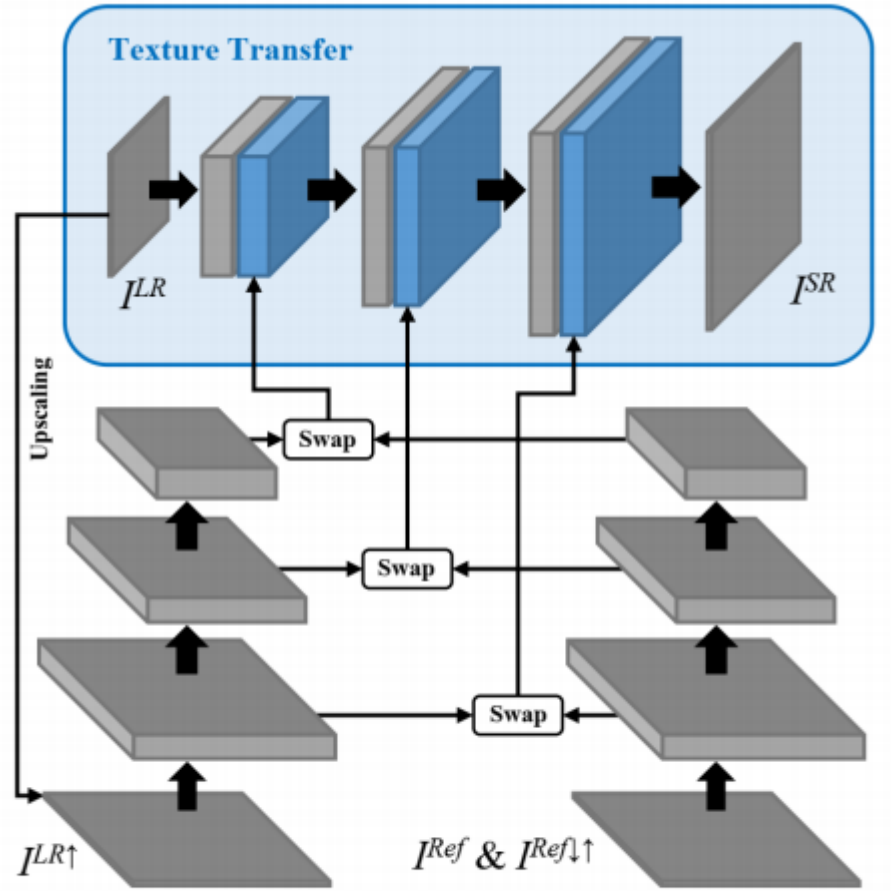

Figure 2: The proposed SRNTT framework with feature swapping and texture transfer.

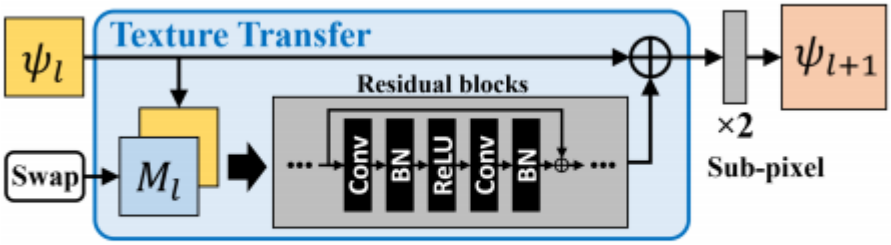

Figure 3: The network structure for texture transfer.

- SRNTT는 기본적으로 4x scale 에 대해 학습하는 모델로 구성했다. 그래서 LR 과 HR는 4배 차이가 나도록 구성해서 데이터셋을 구성해야된다. HR 이미지의 크기는 160×160 으로, LR도 그에 맞춰서 4배 줄여주면 된다.
- 구조는 생각보다 심플하다. 그림 3과 같이 Residual blocks 을 통해 SR 학습이 진행이 된다. 마지막에는 ESPCN에서 제안한 Sub-Pixel Convolution 을 적용해서 각

blocks 이후에 2배씩 upscale 되도록 해준다. 최종적으로 Residual blocks 를 2개를 구성해서 upscale을 2번 적용되어 HR이 나오도록 한다.

- Upscale 전에 Residual blocks에서 나온 feature에 더해지는 것이 있는데 바로 Texture feature다.
- 그림 1을 보자, **Feature Swapping**이라는 메카니즘이 등장한다. 이것이 존재하는 이유는 Reference 이미지의 feature(VGG19로부터 추출된)를 바로 보내는 것이 아니기 때문이다. 먼저 target과 reference 두 이미지의 feature들을 뽑고 특정 크기의 Patch로 둘 다 분할한다(더 작은 사이즈 이미지들로 자른다고 생각해라). 둘 다 같은 크기의 이미지였다면, 동일한 n개의 조각이 나올거고, 그렇지 않다면 n개와 m개의 조각들이 서로 나올 것이다. 뭘 기준으로 둘지는 상관없지만 편의상 원본 이미지를 기준으로하고 원본 이미지의 쪼개진 n개의 feature Patch에서 i번째 patch을 지정하자. 그리고 i번째 patch을 가지고 Reference 이미지의 쪼개진 m개의 feature Patch 전부에 대해 하나씩 유사도 score 를 계산한다. 이 과정을 수식으로 전개한 식은 (1), (2) 로 진행된다.

$$s_{i,j} = \left\langle P_i(\phi(I^{LR\uparrow})), \frac{P_j(\phi(I^{Ref\downarrow\uparrow}))}{\|P_j(\phi(I^{Ref\downarrow\uparrow}))\|} \right\rangle, \qquad (1)$$

$$S_j = \phi(I^{LR\uparrow}) * \frac{P_j(\phi(I^{Ref\downarrow\uparrow}))}{\|P_j(\phi(I^{Ref\downarrow\uparrow}))\|}, \qquad (2)$$

- i번째 원본 feature Patch에 대해 m개의 유사도 Score가 계산될 텐데 그 중 제일 Score가 높은 j번째 Reference feature Patch를 뽑아둔다. 그렇게 n개의 featrue Patch를 다 돌면, 제일 유사한 Reference feature Patch도 n개가 뽑힐텐데, 이를 순서대로 이어붙여 쪼개기 전의 사이즈와 동일한 새로운 feature로 만들어낸다. 정리하자면, 이제 이 새로운 feature는 원본이나 Reference 이미지의 feature가 아닌, 특정 영역마다 제일 유사한 texture 정보가 짜집기 된 feature가 된 것이다.

$$P_{\omega(x,y)}(M) = P_{j^*}(\phi(I^{Ref})),\ j^* = \arg\max_j S_j(x,y), \quad (3)$$

- 보통은 이런 과정을 VGG19에서 relu1_1, relu2_1, relu3_1 총 3 단계에서 다 해줘야되는데, 위에 서술한 것처럼 가뜩이나 반복 연산으로 꽤 연산량이 증가했는데 3번이나 해야되서 여러모로 좋지가 않다. 따라서 각 Patch의 유사도 계산은 relu3_1에서만 해

주고 그 과정에서 우리가 알 수 있는 Patch의 중심좌표(세번째 CNN blocks 기준으로)를 바탕으로 relu1_1, relu2에서의 Patch 중심좌표도 추적한다. 이때는 유사도 계산은 안하고 그냥 바로 추적한 중심좌표의 Reference Feature의 Patch들만 뽑아 마찬가지로 각각 새로운 feature로 만든다.(더 깊은 영역에서 뽑힌 feature로 이미 유사도 계산을 했으니 상위 blocks의 feature들도 비슷할거다 라는 취지인듯)

- 그렇게 각 단계마다 총 세 개의 feature가 만들어지는데 이 feature를 메인 네트워크에서 나온 feature에 concatenate 하는 식으로 넘겨주면 된다.

$$I^{SR} = \mathrm{Res}\left(\psi_{L-1} \| M_{L-1}\right) + \psi_{L-1} \qquad (5)$$

- 글로 설명하니 복잡한데 논문을 직접 보면 좀 더 쉽게 설명되니 관심있는 분들은 직접 정독하는 것을 추천한다.
- VGG19에 Ref&원본 이미지를 넣기 전에 주의해야 될 점은 Reference 이미지는 기본적으로 고해상도이지만 VGG19에 넣는 이미지는 LR 이미지를 Reference와 동일한 크기로 upscale(논문에선 bicubic Interpolation)한 이미지를 넣는 것이다. 따라서 한 쪽이 상대적으로 더 고해상도이기 때문에 LR 이미지의 frequency band에 맞게 끔 Reference 이미지도 blurry하게 만든다고 한다. (4x downsampling 후 다시 4x upsampling).
- 그럼 둘 다 저해상도 이미지로 진행되니깐 이를 막기 위해 실제로는 같은 크기의 고해상도 Reference 이미지와 blurry한 Reference 이미지, 그리고 원본 LR 이미지(Upsampled) 세 개를 넣어주고, 유사도 계산은 blurry 한 Ref와 원본 LR 끼리 진행하고 거기서 얻은 Patch의 중심좌표를 고해상도 Ref 이미지의 feature에 적용해 texture 정보를 뽑아서 전달하는 것이다.

Training

보통 SISR은 Reconstruct Loss(대표적으로 MSE) 이나 Adversarial Loss(GAN), 간혹 Perceptual Loss 를 가지고 진행되는데, SRNTT의 경우엔 이 3가지와 Style Transfer에서 사용되는 Gram Matrix를 이용한 Texture loss 총 4가지를 다 사용한다.

$$\mathcal{L}_{tex} = \sum_{l} \lambda_l \left\| Gr\left(\phi_l(I^{SR}) \cdot S_l^*\right) - Gr\left(M_l \cdot S_l^*\right) \right\|_F, \qquad (6)$$

- λ는 레이어 *l* 의 feature size에 대응하는 normalization factor다.

$$\mathcal{L}_{rec} = \left\| I^{HR} - I^{SR} \right\|_1, \tag{7}$$

- Reconstruct loss는 대표적으로 MSE를 사용하지만, 이번 논문에서는 l1-loss을 사용했다고 한다.

$$\mathcal{L}_{per} = \frac{1}{V} \sum_{i=1}^{C} \left\| \phi_i(I^{HR}) - \phi_i(I^{SR}) \right\|_F, \tag{8}$$

- Perceptual Loss는 Pretrained model 단에서 추출된 feature가 같게끔 하는 loss 이다. 각 픽셀끼리 계산한 다는 점은 Reconstruct loss랑 같아 보이지만 결과 output에 대한게 아니라 feature 끼리 계산하는 것이 Perceptual loss 이다. VGG19의 relu5_1 layer의 feature들로 계산해준다. V와 C는 볼륨(width * heigth)과 채널 수다. i는 i번째 채널의 feature maps이다.

$$\mathcal{L}_{adv} = -\mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})], \tag{9}$$

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})], \tag{10}$$

- Adversarial loss는 선명한이나 현실감을 올려주는데 효과가 크다. 그 중에서도 안정적인 결과를 얻을 수 있는 WGAN-GP 방식을 적용했다고 한다. WGAN의 Wasserstein distance 가 기본적으로 *l1-norm*을 기반으로 하는데, 본 논문도 마침 Reconstruct loss에도 *l1-norm*을 사용하기 때문에 선택했다고 한다. *D*는 1-Lipschitz functions로 구성되었다고 한다.

Details

- 아까도 언급했듯이 VGG19 를 pretrained model로 사용했다.
- Feature layer로는 relu1_1, relu2_1, relu3_1으로 정했다.
- 최종 Loss는 네 가지 loss : *L_rec, L_per, L_adv, L_tex* 에 가중치를 줘서 더한다. 각각 1, 1e-4, 1e-6, 1-e4의 가중치를 부여해준다.
- 처음 2 epochs 동안에는 오직 *L_rec*만 가지고 학습시키고 그 이후로부터 20 epochs 동안에는 네 가지 loss 전부 사용해서 학습한다.

Dataset

- 논문에서는, RefSR을 위한 데이터셋이 따로 존재하지 않았기 때문에 직접 Reference 를 포함해서 데이터셋을 준비했다고 한다.
- 학습에 사용된 데이터셋은 CUFED 로, 일상을 담은 1,883개의 사진들을 포함하고 있다고 한다.
- SIFT를 기반으로 한 유사도 정도를 가지고서 Reference의 정도를 총 4가지 level로 설정해서 한 이미지에 대한 Reference 이미지를 4가지씩 만들었다. 또한, 원본 HR의 이미지로부터 랜덤하게 160×160 크기의 Patch들을 Crop 했다고 한다.
- 최종적으로는 13,761 개의 학습 데이터셋이 만들어졌다고 한다. 게다가 논문 저자는 학습과 평가에 사용한 데이터셋 전부를 공개했다. ~~(아주 바람직하다.)~~
- 평가는 CUFED5 test set, Sun80, Urban 100 으로 진행했다.

Table 1: PSNR/SSIM comparison of different SR methods on three datasets. Methods are grouped by SISR (top) and RefSR (bottom) with their respective best numbers in bold.

Algorithm	CUFED5	Sun80 [33]	Urban100 [17]
Bicubic	24.18 / 0.684	27.24 / 0.739	23.14 / 0.674
SRCNN [5]	25.33 / 0.745	28.26 / 0.781	24.41 / 0.738
SelfEx [16]	23.22 / 0.680	27.03 / 0.756	24.67 / 0.749
SCN [37]	25.45 / 0.743	27.93 / 0.786	24.52 / 0.741
DRCN [22]	25.26 / 0.734	27.84 / 0.785	25.14 / 0.760
LapSRN [23]	24.92 / 0.730	27.70 / 0.783	24.26 / 0.735
MDSR [25]	**25.93 / 0.777**	**28.52 / 0.792**	**25.51 / 0.783**
ENet [30]	24.24 / 0.695	26.24 / 0.702	23.63 / 0.711
SRGAN [24]	24.40 / 0.702	26.76 / 0.725	24.07 / 0.729
SRNTT-ℓ_2 (SISR)	25.91 / 0.776	28.46 / 0.790	25.50 / 0.783
Landmark [39]	24.91 / 0.718	27.68 / 0.776	—
CrossNet [41]	25.48 / 0.764	28.52 / 0.793	25.11 / 0.764
SRNTT-ℓ_2	**26.24 / 0.784**	**28.54 / 0.793**	**25.50 / 0.783**
SRNTT	25.61 / 0.764	27.59 / 0.756	25.09 / 0.774

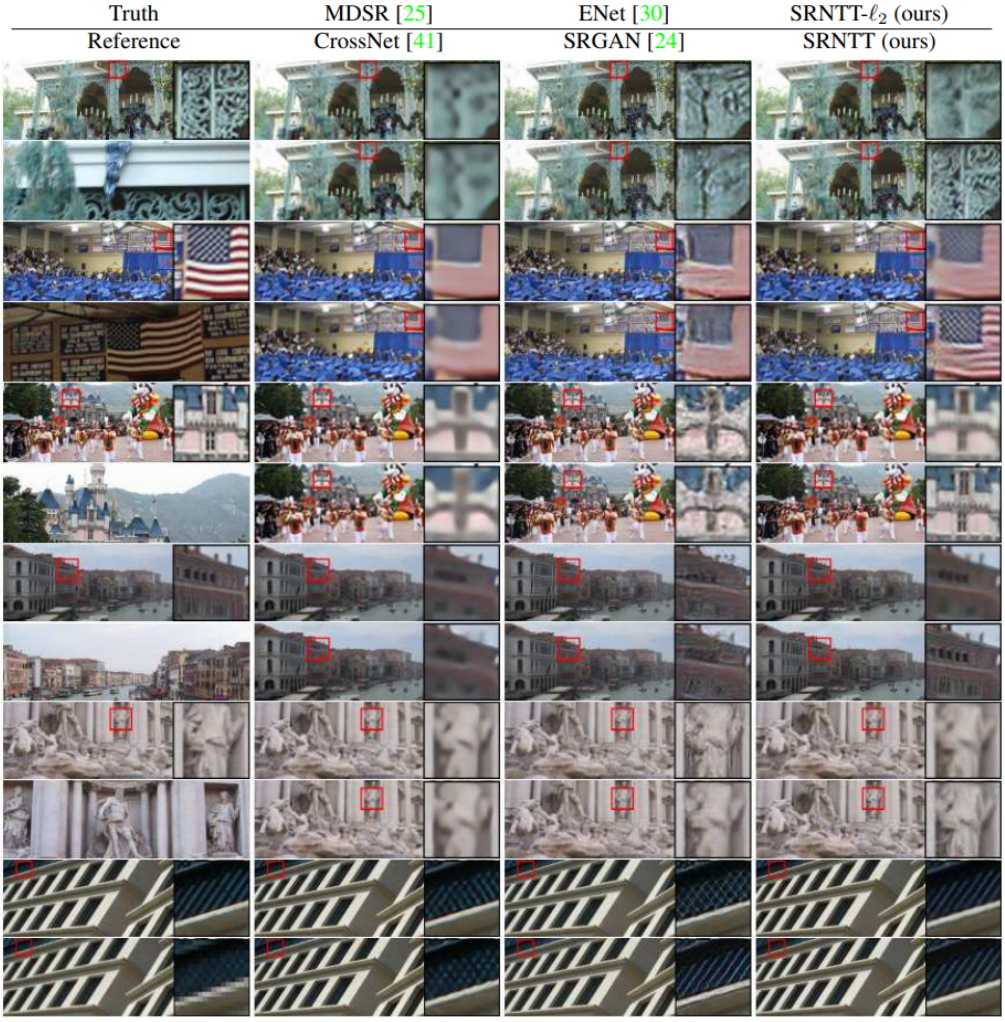

Figure 5: Visual comparison among different SR methods on CUFED5 (top three examples), Sun80 [33] (the forth and fifth examples), and Urban100 [16] (the bottom example whose reference image is the LR input).

Table 2: PSNR/SSIM at different reference levels on CUFED5 dataset. PM indicates if patch-based matching is used; GAN indicates if GAN and other perceptual losses are used.

	PM	GAN	HR (warp)	L1	L2	L3	L4	LR
CrossNet [41]			25.49 / .764	25.48 / .764	25.48 / .764	25.47 / .763	25.46 / .763	25.46 / .763
SRNTT-ℓ_2	✓		29.29 / .889	**26.15 / .781**	**26.04 / .776**	**25.98 / .775**	**25.95 / .774**	**25.91 / .776**
SRNTT-flow		✓	25.82 / .801	24.64 / .743	24.22 / .723	24.15 / .719	24.05 / .714	25.50 / .756
SRNTT	✓	✓	**33.87 / .959**	25.42 / .758	25.32 / .752	25.24 / .751	25.23 / .750	25.10 / .750

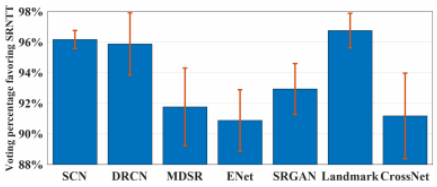

Figure 6: The user study result. SRNTT is compared to each algorithm along the horizontal axis, and the blue bars indicate the percentage of users favoring SRNTT results.

Table 3: PSNR of using different VGG layers for feature swapping on different reference levels.

Layer	relu1	relu2	relu3	relu1/2	relu1/2/3
HR	28.39	28.66	24.83	30.39	33.87
L1	24.76	24.91	24.48	25.05	25.42
L2	24.68	24.86	24.22	25.00	25.32
L3	24.64	24.80	24.39	24.94	25.24
L4	24.63	24.79	24.45	24.92	25.23

Valuable Relative Works

- ESPCN