

# Deep Learning을 이용한 Image Super Resolution

국민대학교 권영훈

December 15, 2016

## 1 연구 배경

Image Super Resolution이란 저화질 이미지를 고화질 이미지로 만드는 방법이다. 컴퓨터 비전분야에서 Image Super Resolution에 대한 많은 연구를 진행해왔다(1).

Deep Learning이 발전하면서 컴퓨터 비전분야에도 많은 문제를 해결하고 있다. 2015년에 발표된 Chao Dong 등의 논문 Image Super-Resolution Using Deep Convolutional Networks(이하 SRCNN)(2)와 2016년에 발표된 Jiwon Kim 등의 논문 Accurate Image Super-Resolution Using Very Deep Convolutional Networks(이하 VDSR)(3)는 Deep Learning으로 Image Super Resolution을 해결하는 방법을 제시했다.

본 연구는 두 논문을 참고하여 Image Super Resolution 문제를 Google의 Tensorflow를 사용하여 해결하고자 한다.

## 2 연구 내용

### 2.1 문제 접근 및 핵심 내용

Image Super Resolution이란 저화질 이미지를 고화질 이미지로 만드는 방법이다. 기존에는 예측 모델(prediction models), 윤곽 기반 방법(edge based methods), 영상 통계 기법(image statistical methods) 그리고 패치 기반 방법(patch based methods) 등으로 문제를 풀어왔다.

Deep Learning을 적용한 알고리즘은 입력의 영상으로 저화질 이미지가 들어오면 Convolutional Neural Network(이하 CNN)를 거쳐 고화질 이미지가 출력이 되는 방법으로 구현한다. 저화질의 입력 이미지가 주어지면 각 단계에서 사용되는 Convolutional 필터를 거쳐 최종적으로 고화질 이미지를 얻는다. CNN을 구현하기 위해서는 CNN 구조를 모델링 해야한다.

SRCNN에서 제시한 CNN 모델은 패치 추출 및 표현단계, 비선형 매핑단계 그리고 복원단계로 구성된다. SRCNN의 Learning 방법은 저화질 이미지가 입력으로 들어왔을 때 바로 고화질 이미지를 출력하는 최적의 파라미터를 구하는 방식이다.

VDSR에서는 SRCNN과 다르게 Residual image를 사용하여 Learning을 진행한다. Residual image는 고화질 이미지와 저화질 이미지의 차를 의미한다. Residual Learning 방법은 결과적으로 더 나은 성능을 보인다.

이 연구에서는 Tensorflow를 사용하여 SRCNN의 조건에 맞게 구현을 하고 VDSR의 Residual Learning 방법을 적용하여 구현한것을 비교한다.

## 2.2 SRCNN의 구조

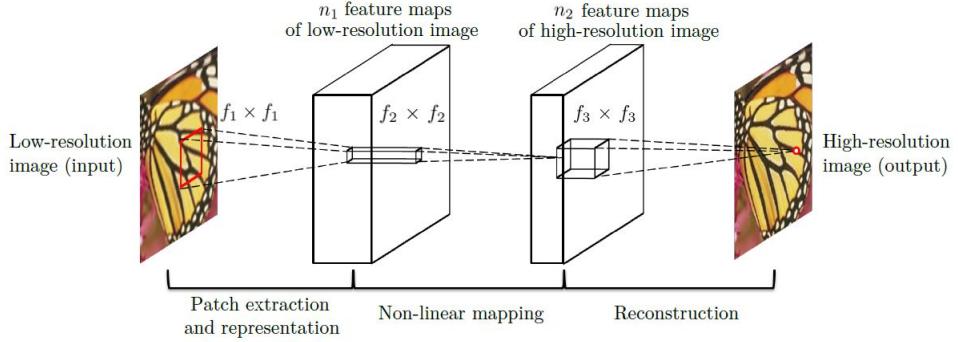


Figure 1: SRCNN 구조 (2).

SRCNN에서 제시한 CNN 모델은 패치 추출 및 표현단계, 비선형 매핑단계 그리고 복원단계로 구성된다. 최적화된 파라미터를 사용하면 입력으로 저화질 이미지가 들어왔을 때 세 단계를 거쳐 고화질 이미지로 변하게 된다.

**첫 번째 단계는 패치 추출 및 표현 단계이다.** 이 단계의 목적은 저화질의 이미지 입력으로부터 특징을 뽑아내어  $n_1$  feature maps로 표현하는 것이다.  $n_1$  feature maps는 입력이미지의 다양한 특징을 뽑아 낸 새로운 이미지 세트라고 볼 수 있다. Convolutional 필터를 사용해 입력 이미지와 컨볼루션 연산을 하여  $n_1$  feature maps를 만든다. 이미지는 *height*  $\times$  *width*  $\times$  *channels*로 나타낸다. 여기서 *channels*는 채널 개수를 의미한다. 1단계 Convolutional 필터는 *channels*  $\times$   $f_1 \times f_1$ 으로 나타내며  $f_1$ 은 Convolutional 필터의 크기를 의미한다. 입력 이미지와  $n_1$ 개의 필터를 컨볼루션 연산을 하여  $n_1$  feature maps를 만든다. 이 연산을 식(1)으로 표현할 수 있다.

$$F_1(X) = \max(0, W_1 * X + B_1) \quad (1)$$

$X$ 는 저화질 입력이미지이다.  $W_1$ 은 1단계 Convolutional 필터의 Weight이다. Weight는 필터 각각의 값을 담은 행렬이다.  $B_1$ 은  $n_1$ 차원 상수벡터이다.  $\max$ 는 Relu를 의미하며 activation function으로 사용된다. 입력으로 들어온 이미지를  $W_1$  필터에 컨볼루션하여  $n_1$  feature maps인  $F_1(X)$ 를 생성한다.  $F_1(X)$ 가 의미하는 것은 패치를 추출하여 표현한 feature maps이다.

**두 번째 단계는 비 선형 매핑 단계이다.** 이 단계의 목적은  $n_1$  차원의 벡터를  $n_2$  차원의 벡터에 비선형적으로 매핑하는 것이다.  $n_1$  feature maps를 Convolutional 필터를 사용하여  $n_2$  feature maps로 표현한다.  $n_1$  feature maps는 *height*  $\times$  *width*  $\times$   $n_1$ 으로 나타낸다. 2단계의 Convolutional 필터는  $n_1 \times f_2 \times f_2$ 로 나타내며  $F_2$ 는 Convolutional 필터의 크기를 의미한다.  $n_1$  feature maps와  $n_2$  개의 2단계 필터를 컨볼루션 연산을 하여  $n_2$  feature maps를 만든다. 이 연산을 식(2)로 표현할 수 있다.

$$F_2(X) = \max(0, W_2 * F_1(X) + B_2) \quad (2)$$

$F_1(X)$ 는  $n_1$  feature maps이다.  $W_2$ 는 2단계 Convolutional 필터의 Weight이다.  $B_2$ 는  $n_2$  차원 상수벡터이다.  $n_1$  feature maps를  $W_2$  필터에 컨볼루션하여  $n_2$  feature maps  $F_2(X)$ 를 생성한다.

세 번째 단계는 복원 단계이다. 이 단계의 목적은 위의 방식의 표현을 종합하여 최종 고해상도 이미지를 생성하는 것이다.  $n_2$  feature maps를 Convolutional 필터를 사용하여 고해상도 이미지로 표현한다.  $n_2$  feature maps는  $height \times width \times n_2$  으로 나타낸다. 3단계의 Convolutional 필터는  $n_2 \times f_3 \times f_3$  로 나타내며  $f_3$ 는 Convolutional 필터의 크기를 의미한다.  $n_2$  feature maps와 3단계 필터를 이미지 채널의 개수만큼 컨볼루션 연산을 하여 고해상도 이미지를 만든다. 이 연산을 식(3)로 표현할 수 있다.

$$F_3(X) = W_3 * F_2(X) + B_3 \quad (3)$$

$F_2(X)$ 는  $n_2$  feature maps이다.  $W_3$ 는 3단계 Convolutional 필터의 Weight이다.  $B_3$ 는 channels의 개수 만큼의 차원을 가진 상수벡터이다.  $n_2$  feature maps를  $W_3$  필터에 컨볼루션하여  $F_3(X)$ 를 생성한다.  $F_3(X)$ 는  $height \times width \times channels$  로 표현되며 여기서 padding은 생략한다.  $F_3(X)$ 가 고화질 이미지가 되도록 하는 파라미터  $W_1, W_2, W_3, B_1, B_2, B_3$ 를 구해야 한다. 최적의 파라미터를 구하기 위해서 Deep Learning을 사용한다.

### 2.3 SRCNN의 Learning 방법

저화질의 입력 이미지를 넣었을 때 고화질의 출력 이미지를 얻으려면 최적화된 파라미터를 구해야 한다. SRCNN에서는 고화질 이미지와 출력 이미지를 비교하여 두 이미지의 차이가 최소가 되는 방향으로 파라미터인  $W_1, W_2, W_3$ 와  $B_1, B_2, B_3$ 를 업데이트 한다.

두 이미지의 차이가 최소가 되는 방향으로 업데이트 하기 위해서 Loss Function을 정의해야 한다. CNN을 Learning 한다는 의미는 Loss Function이 최소의 값을 갖도록 파라미터를 업데이트 하는 것이다. SRCNN의 Loss Function은 다음과 같이 정의할 수 있다.

$$L(\theta) = 1/n \sum_{i=1}^n \|F(X_i; \theta) - Y_i\|^2 \quad (4)$$

$\theta$ 는 각각의 파라미터  $W_1, W_2, W_3, B_1, B_2, B_3$ 를 의미한다.  $n$ 은 트레이닝 데이터의 이미지 개수를 의미한다.  $F(X_i; \theta)$ 는 식(3)의  $F_3(X)$ 를 의미하며 CNN을 거친 출력 이미지를 의미한다.  $Y_i$ 는 고화질 이미지를 의미한다. 평균 제곱근 오차(Mean Squared Error)를 적용하여 Loss function을 정의하였다.

### 2.4 VDSR의 Residual Learning 방법

VDSR의 Learning 방법은 Residual 이미지와 출력 이미지를 비교하여 두 이미지의 차이가 최소가 되는 방향으로 파라미터를 업데이트 하는 것이다. 그림 2의 (c)는 Residual 이미지를 나타낸다. Residual 이미지는 고화질의 원본 이미지(그림 2의 (a))와 저화질의 입력 이미지 (그림 2의 (b))의 차를 의미한다. VDSR에서 얻고 싶은 출력 이미지는 최적의 Residual 이미지가 되는 것이다. 고화질 이미지는 입력 저화질 이미지와 최적의 Residual 이미지를 더하여 만들어 진다.



Figure 2: butterfly의 원본, 입력, Residual 이미지

Residual 이미지를 Learning 한다는 것은 자세한 정보를 주어지고 Learning 하는 것과 같은 효과가 있다. VDSR의 Loss Function은 다음과 같이 정의한다.

$$L(\theta) = 1/n \sum_{i=1}^n \|F(X_i; \theta) - r\|^2 \quad (5)$$

$r$ 은 Residual 이미지(고화질의 원본 이미지  $Y_i$  - 저화질의 입력 이미지  $X_i$ )를 의미한다.  $F(X_i; \theta)$ 는 저화질 이미지를 입력으로 주었을 때 CNN을 거친 출력 이미지가 된다. 이 출력 이미지는 Reidual 이미지를 의미하는 것이고 Learning을 통해 최적화된 Residual 이미지를 구한다. 최적화된 Residual 이미지와 입력 이미지를 더하면 고화질 이미지가 된다.

## 2.5 Training Data

CNN을 트레이닝 하기위해 적합한 Training Data를 만들어야 한다.

RGB 채널을 YCbCr 채널로 변경한다. SRCNN논문에서 RGB 채널에 비해 YCbCr이 더 좋은 결과를 얻기 때문에 YCbCr채널을 사용한다(2). 변환 중 Overflow가 생기지 않게 하기위해 최소 최대 범위를 지정한다.

Scale 배율 만큼 줄이고 다시 Scale 배율 만큼 늘려 저화질 입력 이미지를 만든다. Scale은 트레이닝 데이터를 만드는 데 임의로 설정할 수 있는 부분이다. 이미지를 줄이고 늘릴 때 Biqubic 알고리즘을 사용한다. VDSR은 여러가지 Scale을 적용한 트레이닝 데이터를 만들었다(3).

각각의 이미지를  $33 \times 33$  부분 이미지로 나눈다. 가로와 세로를 14픽셀씩 띄워 가면서 부분 이미지를 생성한다. 91개 이미지를 트레이닝 데이터로 사용한다. 91개의 이미지를 각각  $33 \times 33$  으로 나누면 21884개의 부분 이미지가 생성된다.

## 2.6 Deep Learning을 하기 위한 조건 설정

CNN을 구성하는 Convolutional 필터의 초기 설정은  $f_1 = 9$ ,  $f_2 = 1$ ,  $f_3 = 5$ 로 한다. Optimize 한 필터를 찾기 위해  $f_2$ 의 값을 1, 3, 5로 변경해가며 결과를 비교한다. Optimizer는 Gradient Descent와 Adam방법을 적용하여 비교한다. Residual 을 적용한 것과 적용하지 않은 Loss Function 을 사용하여 각각 Learning 하고 비교한다. SRCNN의 구조에서 feature maps의 차원을 결정하는  $n_1 = 64$ ,  $n_2 = 32$ 로 설정한다. 편의상 모든 Learning Rate는 0.00001로 통일한다. 21884개의 부분 이미지를 128만큼의 mini-batch를 적용하여 트레이닝 한다.

## 2.7 Test 단계

Test 데이터 만드는 방법은 다음과 같다. 3배 만큼 줄이고 다시 3배 만큼 늘려 저화질 이미지를 생성한다. RGB채널을 YCrCb 채널로 변경한다. 전체 이미지를 트레이닝 데이터로 생성한다. 트레이닝된 CNN에 Test 데이터를 적용시킨다.

## 3 연구 결과

### 3.1 SRCNN 구현 결과

3배의 Scale을 적용하여 입력 이미지를 만들고 SRCNN을 테스트한 결과이다.

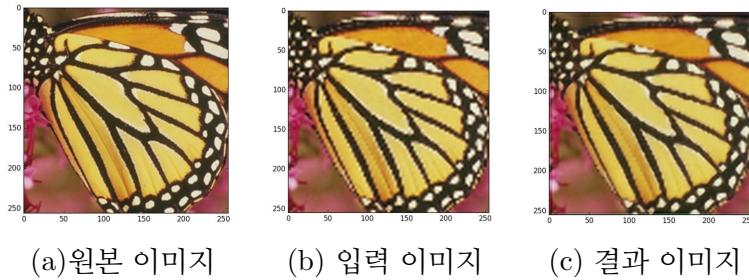


Figure 3: SRCNN 원본, 입력, 결과 이미지

AdamOptimizer 사용 200만번 Learning 한 결과 Cost는 0.000348가 되었다. 입력 이미지와 결과 이미지를 비교해 보면 개선 되었다는 것을 알 수 있다. 원본 이미지와 결과 이미지를 비교해 보면 많은 차이가 있다.

### 3.2 Optimizer 결과 비교

GradientDescentOptimizer 와 AdamOptimizer의 비교를 해본다. GradientDescent는 함수의 극소 점을 찾기 위해 Gradient 반대 방향으로 이동해 가는 방법이다. Adam은 A Method for Stochastic Optimization으로 최근 가장 많이 쓰이는 Optimizer이다 (4). 다음은 가로축 epoch 수 행횟수와 세로축 Cost 결과를 나타낸 그래프이다.

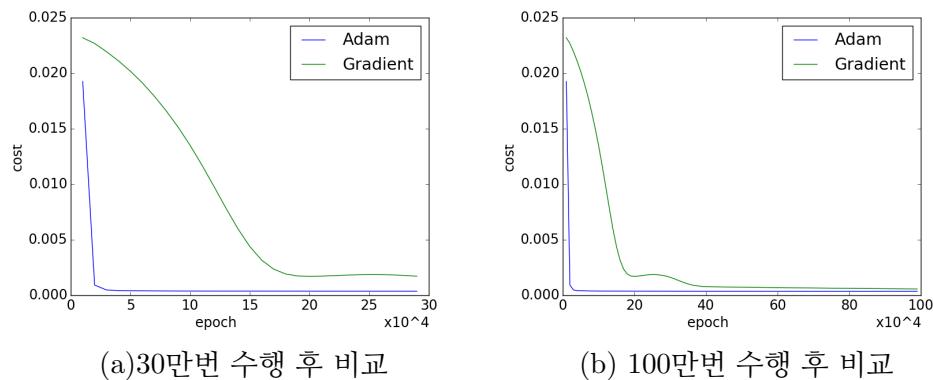


Figure 4: GradientDesent와 Adam Optimizer 비교 그래프

Cost가 작을수록 원본 이미지와 가까운 결과를 나타낸다. 그림 4의 (a)를 보면 천천히 감소하는 Gradient에 비해 Adam은 급격히 감소하는 것을 알 수 있다. AdamOptimizer를 사용하는것이 더 빨리 최저점에 도달하기 때문에 이 후의 Optimizer는 AdamOptimizer를 선택했다.

### 3.3 SRCNN vs SRCNN + Residual 결과 비교

기존의 SRCNN의 조건을 그대로 사용한 결과와 Residual Learning을 적용한 결과를 비교하였다.

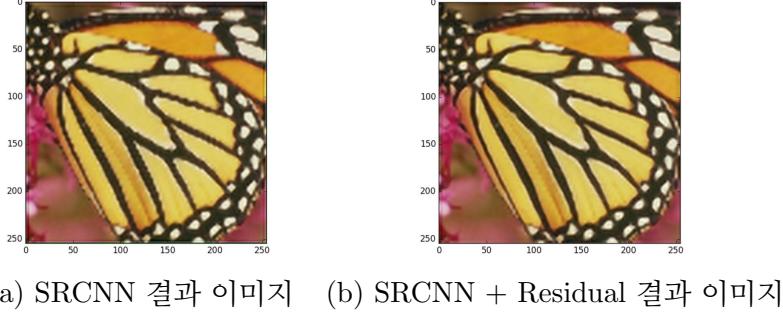


Figure 5: SRCNN과 SRCNN + Residual 결과 이미지

그림 5의 (a)에 비하여 Figure 4의 (b)가 더 부드러운 이미지 처럼 보인다. 다음은 가로축 epoch 수행횟수와 세로축 Cost 결과를 나타낸 그래프이다.

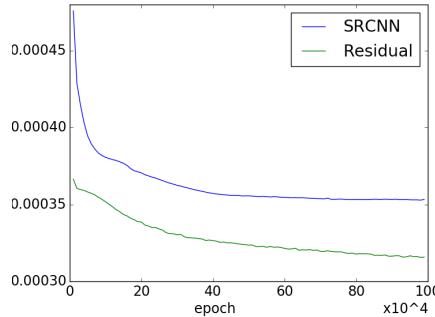
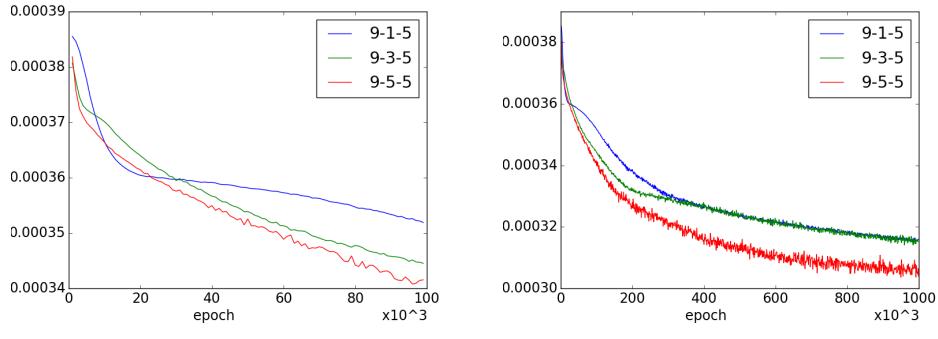


Figure 6: SRCNN과 Residual Learning 각각 100만번 수행 후 Cost

그림 6를 보면 Residual을 적용한 경우에서 Cost가 더 낮게 측정되었다. SRCNN 200만번 Learning 한 결과 Cost는 0.000348이 되었다. SRCNN + Residual 100만번 Learning 결과 Cost는 0.000315이 되었다. 결과적으로 Residual을 적용하면 더 좋은 성능을 보이는 것을 알 수 있다.

### 3.4 Convolutional 필터 9-1-5 vs 9-3-5 vs 9-5-5 결과 비교

SRCNN의 논문에서는 Optimize한 필터의 사이즈를 찾으려 노력했다. Convolutional 필터 f2의 사이즈를 1, 3, 5로 변경해가며 비교해 보았다.



(a) 10만번 수행 후 결과

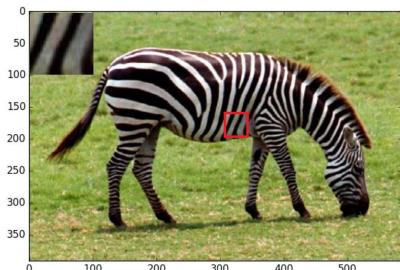
(b) 100만번 수행 후 결과

Figure 7: 두 번째  $W_2$  필터의 크기에 따라 달라지는 결과

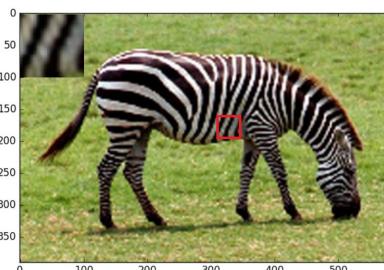
그림 7의 (a)를 보면 9-1-5의 Cost는 빠르게 감소하나 일정 범위에서 정체되는 것을 볼 수 있다. 9-5-5는 Cost가 꾸준히 감소하는 것을 볼 수 있다. 그림 7의 (b)를 보면 100만 번 수행 후 9-1-5와 9-3-5는 비슷한 결과를 보이지만 9-5-5는 더 낮은 Cost가 측정되었다. 필터의 사이즈에 따라 다른 결과를 보이는데 더 많은 실험을 통해 Optimizer한 필터를 정의 찾을 수 있을 것이다.

### 3.5 구현 예시

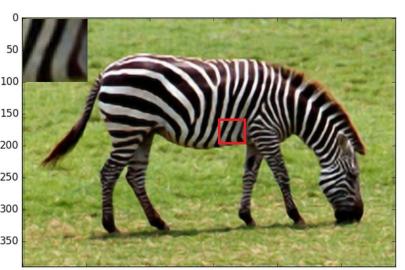
SRCNN + Residual의 9-5-5를 적용한 결과의 예시



(a) 원본 이미지



(b) 입력 이미지



(c) 결과 이미지



(a) 원본 이미지



(b) 입력 이미지



(c) 결과 이미지

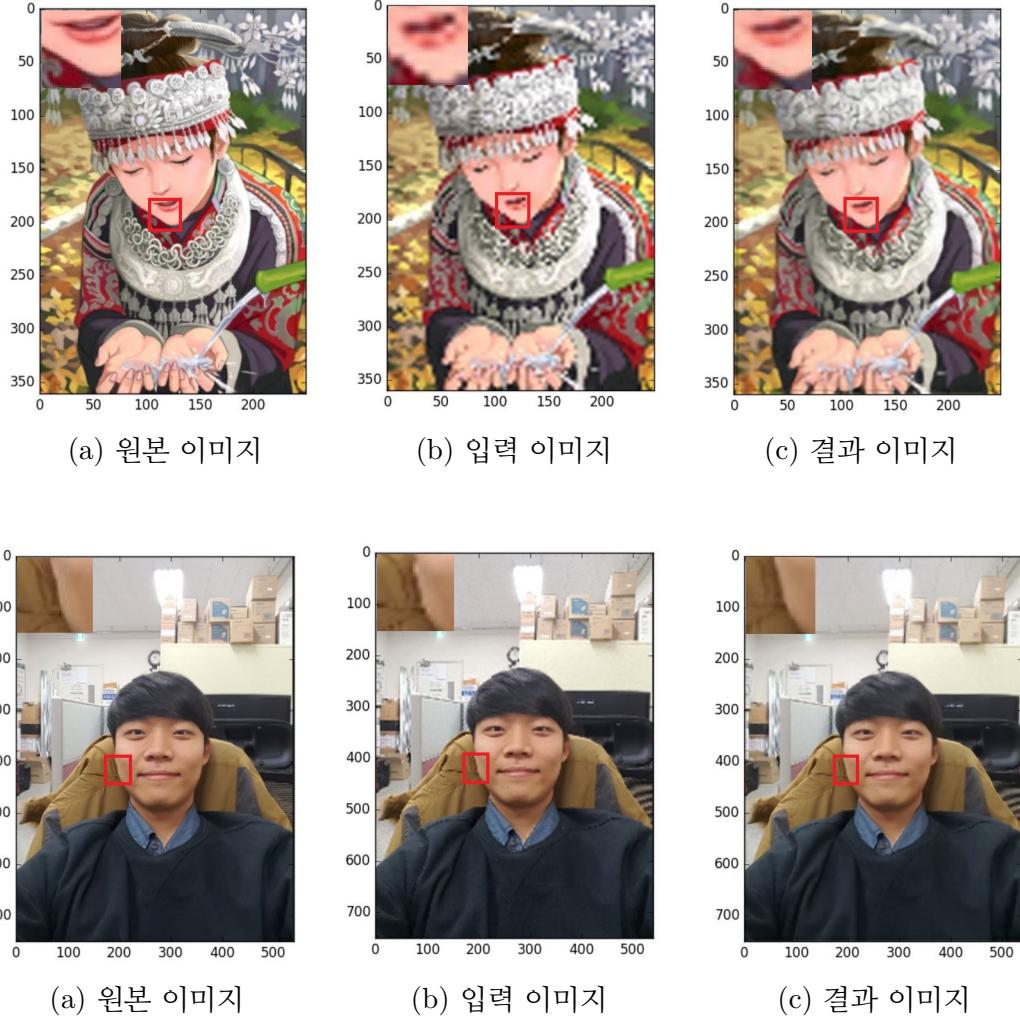


Figure 8: SRCNN Residual Result Examples

## 4 결론

한 학기동안 Deep Learning의 원리를 이해하고 Tensorflow로 구현을 했다. Tensorflow의 튜토리얼을 진행하면서 Machine Learning 알고리즘을 어떻게 구현 할 수 있는지 경험을 쌓았다. Image Super Resolution을 구현하며 많은 시행착오를 겪었다. 여러가지 방법을 사용하여 더 나은 결과를 얻기 위해 노력하였다.

시간이 더 주어진다면 지금의 결과보다 더 개선된 방법으로 Image Super Resolution을 구현할 수 있을 것이다. 먼저 VDSR에서 제시한 더 깊은 레이어를 쌓아 트레이닝 한다. 현재 Convolutional Layer는 3개만 사용하고 있는데 VDSR에서는 20개를 사용하여 트레이닝 하였다. 최적화된 Layer의 개수를 찾아가며 트레이닝 한다면 더 좋은 결과를 얻을 수 있을 것이다. 또한 VDSR에서는 Scale 을 다르게 적용하여 트레이닝 데이터를 만든다. 이렇게 하면 더 많은 조건의 데이터를 학습하게 되어 새로운 데이터를 테스트할 때 성능이 더 좋을 것이다.

결과적으로 Deep Learning으로 구현하면 기존의 알고리즘 보다 더 나은 성능을 보인다. 개선된 Image Super Resolution을 사용하여 일상생활에 응용할 수 있는 애플리케이션을 만들 수 있을 것이다. 저화질의 이미지를 고화질의 이미지로 변환하고 싶은 어떤 이미지라도 적용할 수 있을 것이다.

## References

- [1] Yang, C.Y., Ma, C., Yang, M.H.: Single-image super-resolution: A benchmark. In: European Conference on Computer Vision, pp. 372–386 (2014).
- [2] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang: Image Super-Resolution Using Deep Convolutional Networks (2015).
- [3] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee.: Accurate Image Super-Resolution Using Very Deep Convolutional Networks (2016).
- [4] Diederik P. Kingma and Jimmy Lei Ba: ADAM: A Method For Stochastic Optimization (2015).

**Contributor :** 국민대학교 컴퓨터공학 교수. 김준호, 비주얼컴퓨팅 랩. 이진우