

CNN을 활용한 Super Resolution Platform

졸업과제 최종 보고서

전기컴퓨터공학부 정보컴퓨터공학전공

팀명 : 선명하조?

201524642 김근식

200924467 박형탁

201524652 황남기

지도교수: 차의영 (인)

목차

1. 과제 목표	3
2. 대상 문제 및 요구조건 분석	3
2.1. 문제 분석	3
2.2. 요구 분석	4
3. 현실적 제약 사항 분석 결과 및 대책	4
3.1. 현실적 제약 사항 분석 결과 및 대책	4
3.2. 사업체 멘토링 반영결과	4
4. 설계 문서	5
4.1. 전체 시스템 흐름도	5
4.2. 사진 및 동영상 화질개선 설계	7
1) 기본 인공신경망	7
2) SRCNN	9
3) VDSR	10
3-1) Residual-learning	12
3-2) Gradient clipping	13
5. 진행사항	14
5.1. 영상 프레임 추출	14
5.2. 이론학습	15
5.3. 웹 페이지 구축	18
6. 추진체계	18
6.1. 졸업과제 수행 단계	18
6.2. 구성원 역할분담	19
7. 추진 일정	19
8. 실험 결과 및 결론	19
8.1. 하이퍼 파라미터 튜닝	20
8.2. 실험 결과 및 결론	30
9. 참고 레퍼런스	32

1. 과제 목표

딥러닝 학습을 통해, 저해상도의 사진 및 동영상을 고해상도로 변환하는 것을 목표로 한다.

2. 대상 문제 및 요구조건 분석

2.1. 문제 분석



< 그림1. 고화질 콘텐츠의 발전과정 >

지난해 5월, 지상파 방송사가 초고화질(4K UHD) 본방송 시대를 선언했으나, 실제로 이용할 수 있는 콘텐츠는 아직도 많지 않은 상황이다. 또한, 이제 막 출시되기 시작한 8K TV에 맞는 영상은 사실상 찾아보기 힘든 상황이다. 현재도 제작비 부담에 짓눌려 있는 지상파의 경우, 정부가 권고한 UHD 프로그램 편성비율(2020년 25%, 2023년 50%, 2027년 100%)은 사형선고나 다름없다.

따라서 디스플레이 기술의 비약적인 발전과는 달리 제한된 고품질 콘텐츠 문제를 해결하기 위해 화질을 자동으로 높이는 인공지능 알고리즘이 필요한 시점이다.

2.2. 요구 분석

1) 실시간 이미지 처리

사진 및 동영상은 프레임 단위로 나누어 실시간 처리를 통해 업스케일 처리 후 반영한다.

2) 실시간 처리 서버 구축

본 시스템에 업로드된 사진 및 동영상은 서버에서 처리 과정 후 분석 과정을 거치며,
처리 시스템에서 데이터 축적을 통해 더 나은 결과 값을 도출 해야 한다.

3) 이미지 처리 성공률

학습된 신경망을 통해 80% 이상의 성공률을 보여야 한다.

3. 현실적 제약 사항 분석 결과 및 대책

3.1. 현실적 제약 사항 분석 결과 및 대책

제약사항	설명	대책
데이터 수집에 대한 문제	양질의 데이터 수집에는 프로젝트 기간보다 많은 시간이 소요될 것으로 예상된다.	이미 존재하는 학습 데이터와 새로 수집한 데이터를 함께 활용한다. 새로운 수집한 데이터는 변환 과정 (각도, 크기 변경) 등으로 데이터를 추가한다.
데이터 학습에 대한 문제	데이터 학습에는 상당량의 하드웨어 자원이 필요할 것으로 예상된다.	LAB에 있는 하드웨어 자원을 활용한다.
품질에 따른 화질 개선 문제	소스의 품질이 떨어질 경우 개선에 어려움이 있을 것으로 예상된다.	일정 수준 이상의 품질을 가진 데이터(이미지, 영상)를 입력 받는다.

3.2. 사업체 멘토링 반영결과

과제 구성에는 큰 문제가 없던 것으로 평가 받았으므로 그대로 진행하였다.

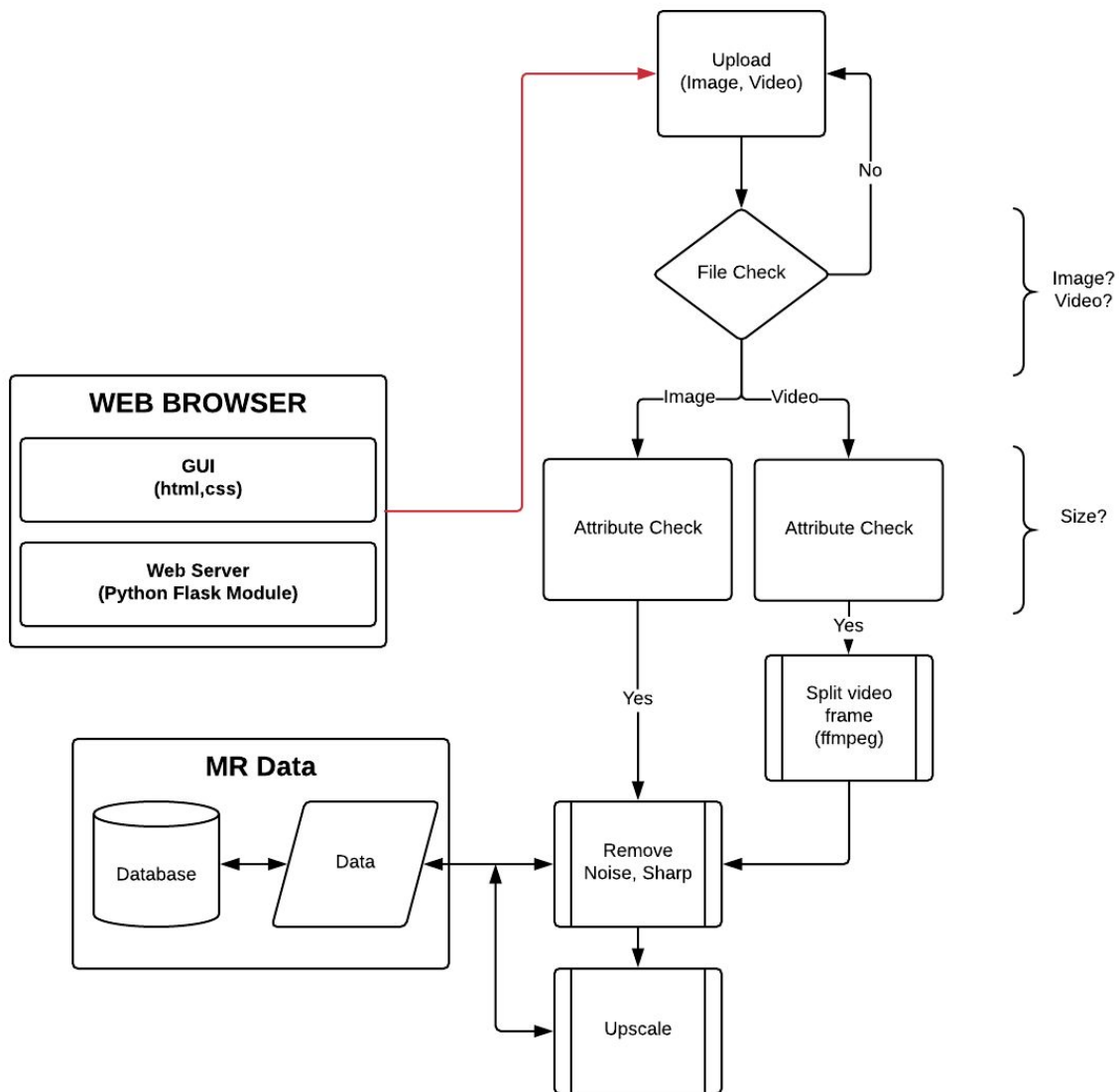
기술 관련으로는 더 많은 고찰이 필요하다는 평가가 있어, 계속해서 조사해본 결과 CNN -> SRCNN -> VDSR의 순서로 기반 기술이 변경됨.

여기에는 Residual Learning, Adjustable Gradient Clipping 등의 기술이 포함되어 있으며, 타당성이 있는 것으로 판단되어 계속 사용하기로 하였다.

상세 기술은 설계 문서에 기술하였다.

4. 설계 문서

4.1. 전체 시스템 흐름도



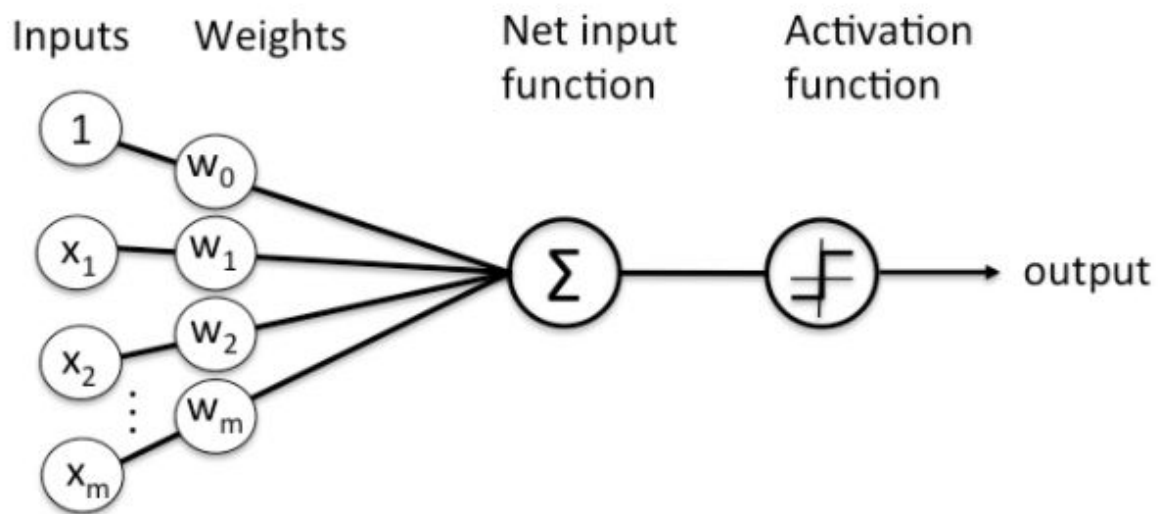
< 그림 2. 전체 시스템 구성도 >

1. 웹을 통해 데이터(사진, 동영상)를 입력 받는다.
2. 이미지의 경우 노이즈 제거, 선명도 증가 등의 처리를 거친 후 업스케일을 진행한다.
3. 영상의 경우 소스를 프레임 단위로 나눈 후 '2'의 처리 작업을 진행한다.
4. 인공지능망 알고리즘을 통해 입력된 데이터(사진, 동영상)를 축적 한다.
5. 최종 결과 데이터(사진, 동영상)를 출력한다.

4.2. 사진 및 동영상 화질개선 설계

1) 인공신경망(Convolutional Neural Network)

인공신경망은 기계학습과 인지과학에서 생물학의 신경망(동물의 중추신경계중 특히 뇌)에서 영감을 얻은 통계학적 학습 알고리즘이다. 인공신경망은 시냅스의 결합으로 네트워크를 형성한 인공 뉴런(노드)이 학습을 통해 시냅스의 결합 세기를 변화시켜, 문제 해결 능력을 가지는 모델 전반을 가리킨다.



< 그림 3. 기본적인 뉴런의 기능모델 >

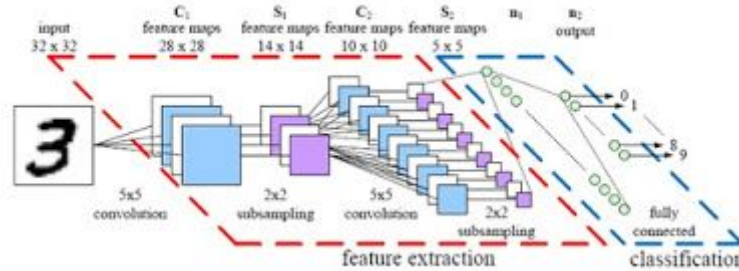
기본적인 인공신경망으로 Feed-Forward Neural Network(FFNN)이 있다. FFNN은 기본적인 연산 과정은 다음과 같다.

- (i) input x 를 입력값으로 인공신경망의 출력값을 계산
- (ii) 가중치 W 를 $y = Wx + b$ 를 통해 계산
- (iii) activation function (ex.. sigmoid, tanh, ReLU)를 적용

(ii)와 (iii)의 과정을 하나의 layer로 볼 수 있는데, 이것들을 여러개 쌓아올린 후 마지막에 (iv) output layer를 쌓아올린 것이 바로 Multilayer Perceptron (MLP)가 되는 것이다. 마지막 layer는 선택지의 수 만큼의 노드를 가지게 되는데 이를 output layer라고 부른다. output layer의 값은 0~1 사이의 값을 가지게 하는게 보통으로, 이를 위해 주로 softmax function을 activation function으로 이용한다.

신경망 학습에는 CNN(Convolutional Neural Network), RNN(Recurrent Neural Network), DBN(Deep Belief Network)등의 방법이 있다.

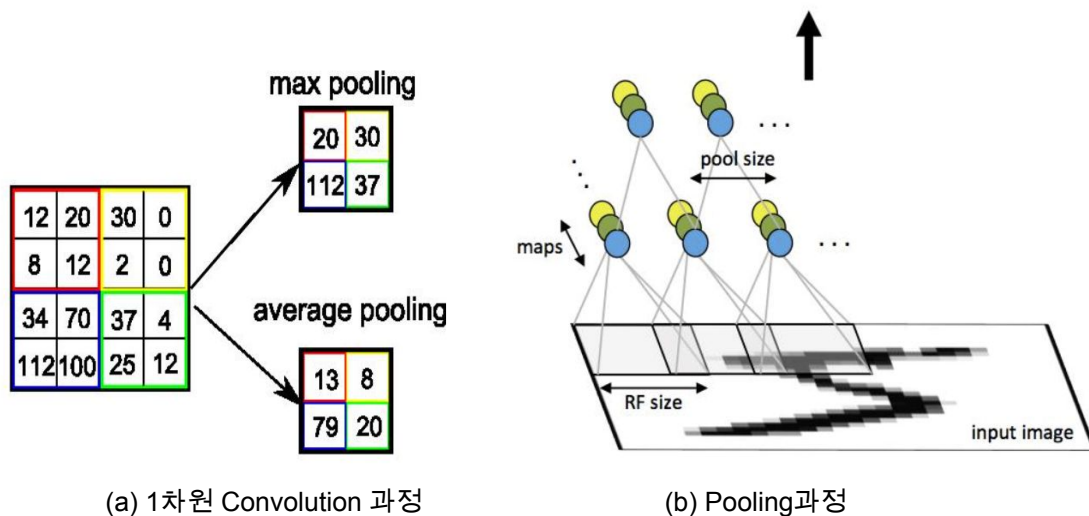
본 시스템에서는 학습된 CNN을 기반으로 저 해상도의 영상이 입력되면 고해상도의 영상을 예측하여 재구성하게 된다.



< 그림 4. CNN의 일반적인 구조 >

CNN은 Convolution과 Pooling 과정을 수행하는데, Convolution이란 커널 행렬로 이미지를 한번 훑어 이미지의 데이터를 다른 형태로 변형 시키는 것이다. 이는 어려운 연산은 아니고 단순히 규칙적으로 원소들끼리 곱셈 연산을 진행 하는 것으로, 왜곡된 이미지를 만들어 feature map을 형성하는 것이다. 이 과정이 끝난 후 CNN은 Pooling 과정을 거치는데, Pooling 과정이란 단순히 말하면 사이즈를 줄이는 과정이라고 볼 수 있다. Convolution이 왜곡시켜놓은 이미지 중 핵심 정보만 간추려 작은 사이즈의 이미지를 만드는 것이다.

또한 CNN은 위 FFNN(그림 3)의 과정이 좀 다른데, 단순히 모든 노드들에 대해 weight를 적용하기보다 receptive fields라고 불리는 일부분에 대해 각각 weight를 적용한다. 즉, 커널 행렬이 학습 가능하다는 것이다. 이 학습 연산은 back propagation을 통해 학습된다.



(a) 1차원 Convolution 과정

(b) Pooling과정

< 그림 5. (a)-(b) CNN의 세부 과정 >

<그림 5-(a)>의 Convolution과 <그림 5-(b)>의 Max-Subsampling은 이미지로부터 한 단계 더 높은 추상화된 정보를 추출한 다음, 그 추상화된 정보에서 가장 중요한 정보만을 남기도록 1/4 크기로 압축, 요약하는 일련의 과정을 보여준다.

CNN은 input data를 2차원 커널 행렬을 통한 Convolution으로 왜곡된 이미지를 얻고나서 activation function을 각각 적용한 후, Pooling으로 영상의 크기를 줄이는 작업을 반복적으로 적용하는 것으로 점차 하나의 일정한 값을 얻어가는 구조라 할 수 있다. 신경망의 마지막 층에는 기본적인 FFNN 하나를 쌓아 CNN의 구조를 완성한다.

2) SRCNN

본 시스템에서는 CNN의 프레임워크로 Tensorflow를 사용하기로 했다. Tensorflow는 대표적인 딥러닝/머신러닝 라이브러리 중 하나로서 고속 컨볼루션 신경망을 구현한 것으로서 널리 사용되고 있다. Tensorflow는 C++ 뿐만 아니라 Python의 인터페이스도 잘 구현되어 있다. Tensorflow를 통해 CNN을 구현하고, GPGPU의 활용을 위한 CUDA 확장기능을 이용하여 학습 속도를 개선시켰다.

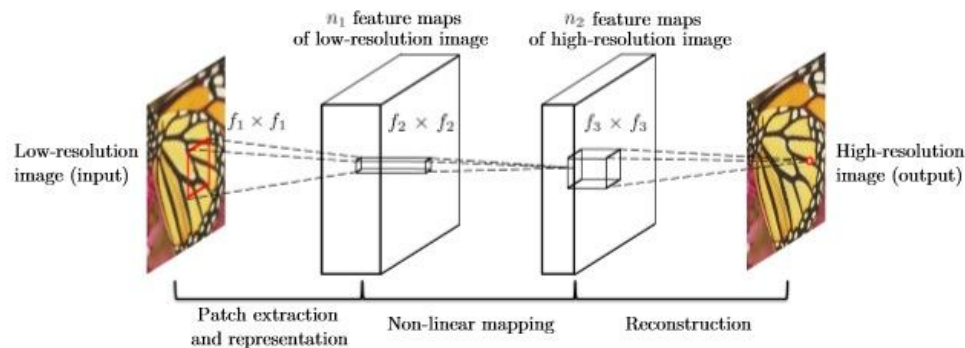


Fig. 2. Given a low-resolution image Y , the first convolutional layer of the SRCNN extracts a set of feature maps. The second layer maps these feature maps nonlinearly to high-resolution patch representations. The last layer combines the predictions within a spatial neighbourhood to produce the final high-resolution image $F(Y)$.

< 그림6. SRCNN 네트워크 구조 >

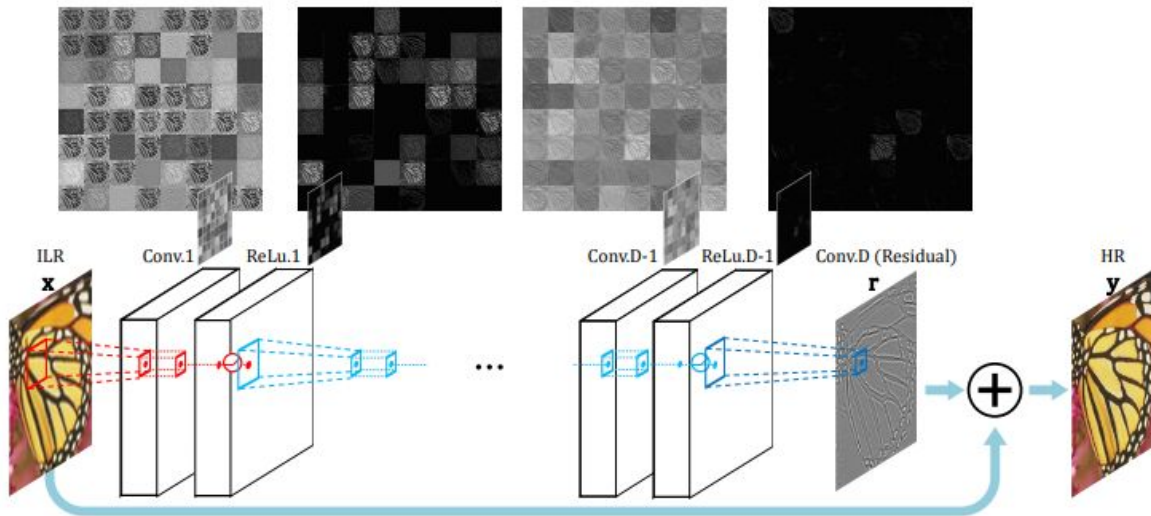
이를 위해 SRCNN(Super-Resolution Convolutional Neural Network)을 고려해보았다. 이는 여타 image super-resolution 모델들에 비해, 속도와 퀄리티 복구에 있어서 월등한 성능이 입증된 모델이다. 하지만 SRCNN은 고사양 컴퓨팅 파워가 필요했기 때문에 실시간 성능이 요구되는 작업에 쓰일 수 없었다. 그래서 SRCNN의 세 가지 한계점을 해결한 VDSR(Very Deep Super-Resolution) 모델을 차용하여 하이퍼 파라미터 튜닝을 통해 우리 문제에 적합한 모델을 만들어냈다.

SRCNN의 대표적인 세 가지 문제점은 다음과 같다. 첫째, 작은 이미지 영역의 문맥에 의존한다. CNN은 인접층의 뉴런 사이에서 로컬 연결 패턴 강화시킴으로써 spatially-local-correlation을 이용한다. 즉, m 층에 있는 히든 유닛은 인풋으로 $m-1$ 층에 있는 유닛들의 부분집합을 입력으로 이용하는데, 이것은 각 히든 유닛은 인풋에 관해서 수용필드 바깥의 변화에 반응하지 않는다는 의미이다. 결국 작은 이미지 영역의 문맥에 의존하게 된다.

둘째, 학습이 너무 느리다는 점이다. SRCNN은 고해상도 이미지를 직접 모델링한다. 저주파 정보(저해상도 이미지에 해당)와 고주파 정보(residual image 또는 image detail information)로 분해될 수 있다. 이 때, 입력 및 출력 이미지는 동일한 저주파 정보 공유하는데, SRCNN에서 이것은 두가지 목적 위해 사용된다. 즉, 입력을 최종 계층으로 운반하고 residuals를 재구성하는 것과 입력단을 끝까지 운반하는 것으로 나눌 수 있다. 후자의 경우 개념적으로 오토 인코더와 유사하다. 이로 인하여 오토 인코더 학습하는데 학습시간 소요되기 때문에 다른 부분(이미지 세부사항) 익히는 수렴속도가 크게 줄었다. 즉, 학습이 느려지는 것이다.

셋째, 네트워크가 단일 규모로만 작동한다는 점이다. SRCNN은 단일 배율 인수에 대해 학습한다. 즉, 지정된 배율에서만 작동한다는 것이다. 따라서 새로운 배율이 요구되면 새로운 모델이 학습되어야 한다. 이는 다중 규모의 SR에 대처하기 위해서는 개별적으로 여러 단일 규모 시스템 구축해야 한다는 문제를 야기하는데, 이것은 비실용적이다.

3) VDSR



< 그림 7. VDSR 처리 과정 >

Context

매우 큰 이미지 영역에 퍼지는 컨텍스트 정보를 사용함으로써 해결한다. 대규모 요소의 경우 작은 패치에 포함된 정보가 세부 복구에 충분하지 않은 경우가 종종 있다. 이 때문에 큰 수신 영역을 사용하는 VDSR의 심층 네트워크는 큰 이미지 컨텍스트를 고려한다.

Convergence

훈련 속도를 향상을 위해 Residual-learning CNN과 매우 높은 학습률을 이용한다. LR 이미지와 HR 이미지는 동일한 정보를 상당 부분 공유하므로 HR 이미지와 LR 이미지의 차이인 Residual-learning을 명시적으로 모델링하는 것이 유리하다. 입력과 출력이 상호 연관성이 높은 경우에 효율적인 학습을 위한 네트워크 구조를 갖는다. VDSR의 경우, 초기 학습 속도는 SRCNN보다 만배나 높다. 이것은 Residual-learning 및 Gradient clipping을 통해 가능하다.

Contribution

요약하면, 본 과제에서는 아주 깊은 심화 CNN을 기반으로 매우 정확한 SR 방법을 제안한다. 아주 깊은 네트워크는 작은 학습률을 사용하면 너무 느리게 수렴한다. 또한 높은 학습률로 수렴 속도를 높이면 exploding gradient가 발생하여, Residual-learning 및 Gradient clipping으로 문제를 해결한다. 또한 단일 네트워크에서 다양한 배율의 SR 문제에 대처할 수 있도록 작업을 확장시킨다.

3-1)Residual learning

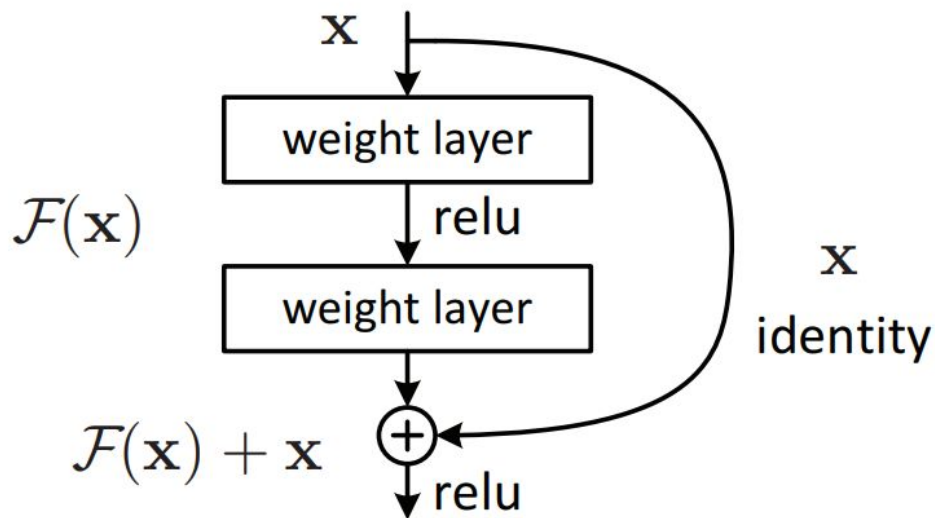


Figure 2. Residual learning: a building block.

< 그림8. Residual-learning >

SRCNN에서 네트워크는 이미지가 폐기되고 학습된 특징만으로 출력이 생성되므로 모든 입력 세부 사항을 보존해야 한다. 많은 weight 층으로 인해 이것은 매우 장기의 메모리를 요구하는 네트워크가 된다. 이러한 이유로 기울기 소실/폭발 문제가 중요 할 수 있다. 이것은 단순히 Residual 학습으로 문제를 해결할 수 있다.

입력 영상과 출력 영상이 대체로 유사하기 때문에 잔여 영상 $r = y - x$ 를 정의한다. 여기서 대부분의 값은 0 또는 작을 것이다. 이 **Residual** 이미지를 예측하려고 할 때, 손실 함수(loss function)는 다음과 같은 형태를 보인다.

$$\frac{1}{2} ||r - f(x)||^2$$

< 그림9. 잔차를 이용한 손실 함수 수식 >

여기서 $f(x)$ 는 네트워크 예측으로, 네트워크에서 이것은 손실 계층에서 다음과 같이 나타난다. 우리의 손실 계층은 Residual 추정, 네트워크 입력 (ILR 이미지) 및 ground truth HR image 세 가지 입력을 받는다. 손실은 재구성된 이미지 (네트워크 입력과 출력의 합)와 ground truth 사이의 유클리드 거리(Euclidean distance)로 계산된다.

3-2) Gradient clipping

학습을 향상시키기 위해 학습률을 높이는 것이 기본 원칙이다. 그러나 단순히 학습률을 높게 설정하면, 기울기 소실/폭발 문제가 발생할 수 있다. 그런 이유로 폭발 gradients를 억제하면서 속도를 최대한 높이기 위해 Adjustable Gradient clipping을 사용한다.

그라디언트 클리핑은 RNN을 학습하는 데 자주 사용되는 기술이다. 그러나 CNN 학습에 그 사용법은 제한되어 있다. 기울기를 제한하는 데에는 많은 방법이 있지만 일반적인 전략 중 하나는 개별 기울기를 미리 정의된 범위 $[-\theta, \theta]$ 로 잘라내는 것이다.

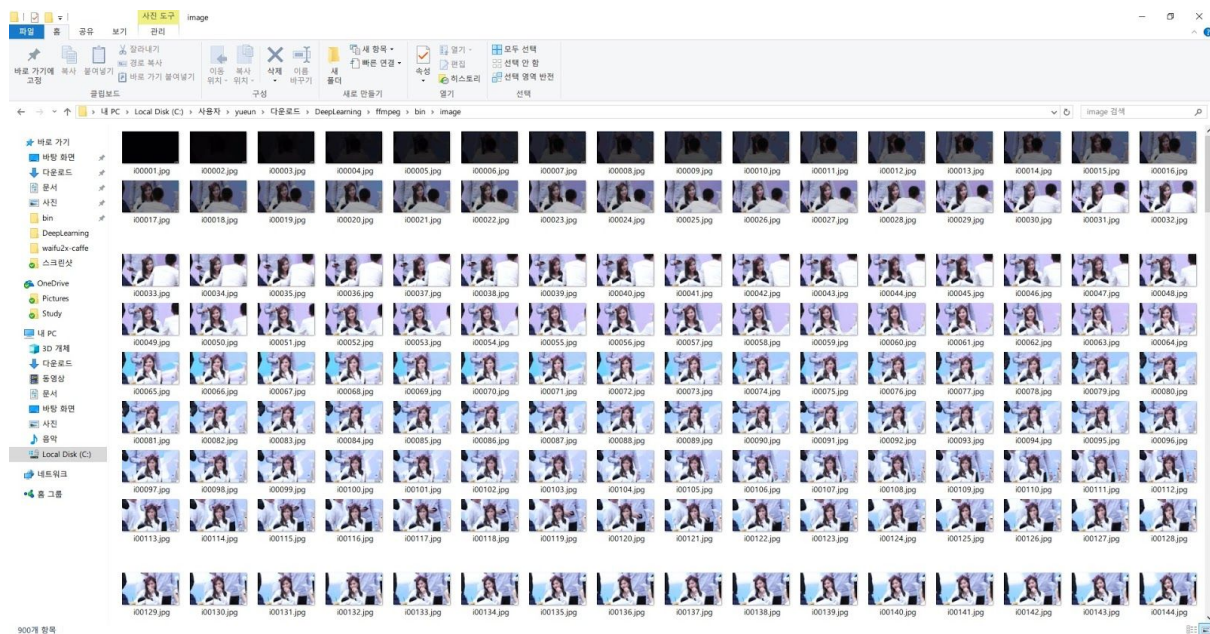
클리핑을 사용하면 기울기가 특정 범위 내에 존재하게 된다. 학습에 일반적으로 사용되는 확률적인 그라디언트 디센트(SGD)를 사용하여 학습 속도를 곱하여 단계 크기를 조정한다. 높은 학습 속도가 사용된다면, 높은 학습 속도 체제에서 폭발하는 그라디언트를 피하기 위해 θ 가 작게 조정될 가능성이 있다. 그러나 학습 속도가 천천히 작아짐에 따라 실제 기울기 (학습 속도가 곱해진 그라디언트)는 0에 가까워지고 학습 속도가 기하학적으로 감소하면, 학습이 기하 급수적으로 많은 반복을 수반할 수 있다.

수렴의 최대 속도를 위해, 기울기를 $[-\theta/\text{감마}, \theta/\text{감마}]$ 로 잘라낸다. 여기서 감마는 현재 학습 속도이다. Adjustable Gradient clipping을 통해 수렴 과정을 매우 빠르게 할 수 있었다. 3계층 SRCNN이 훈련하는 데 며칠이 걸리지만 VDSR 원 저자의 20층 네트워크 학습은 4시간 이내에 완료되었다.

5. 진행 사항

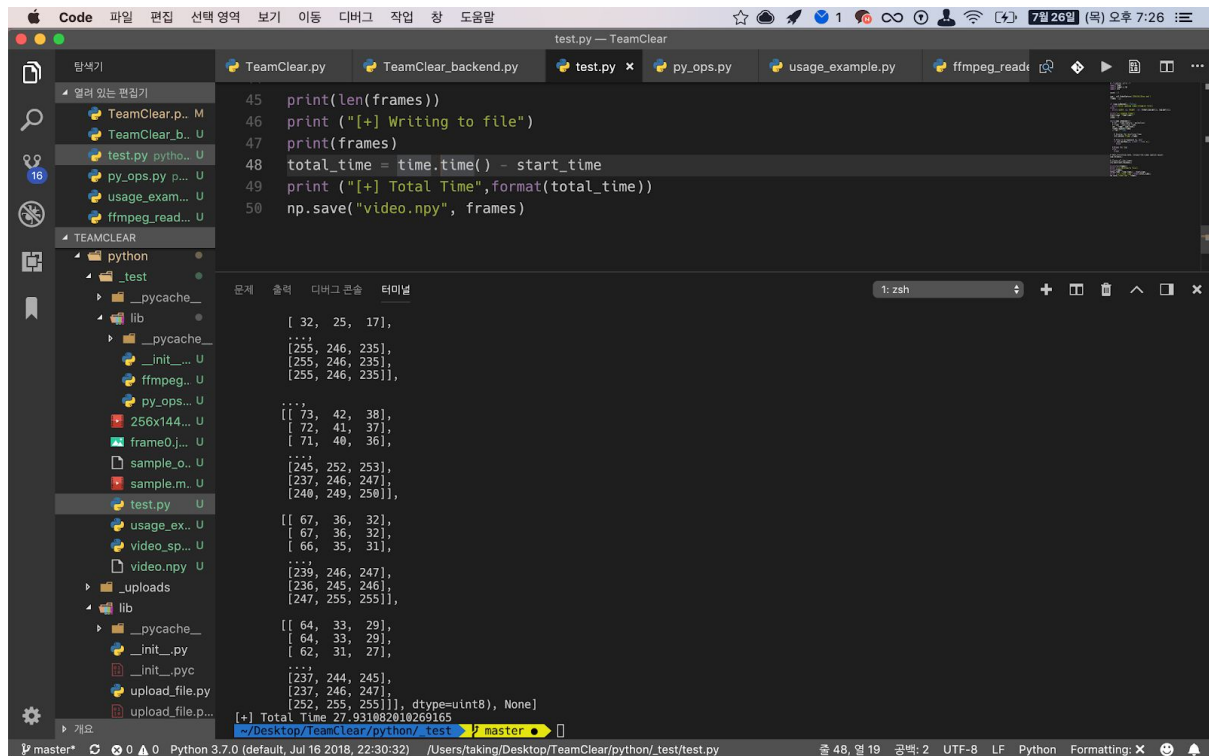
5-1) 영상 프레임 추출

1) 이미지 추출 방식



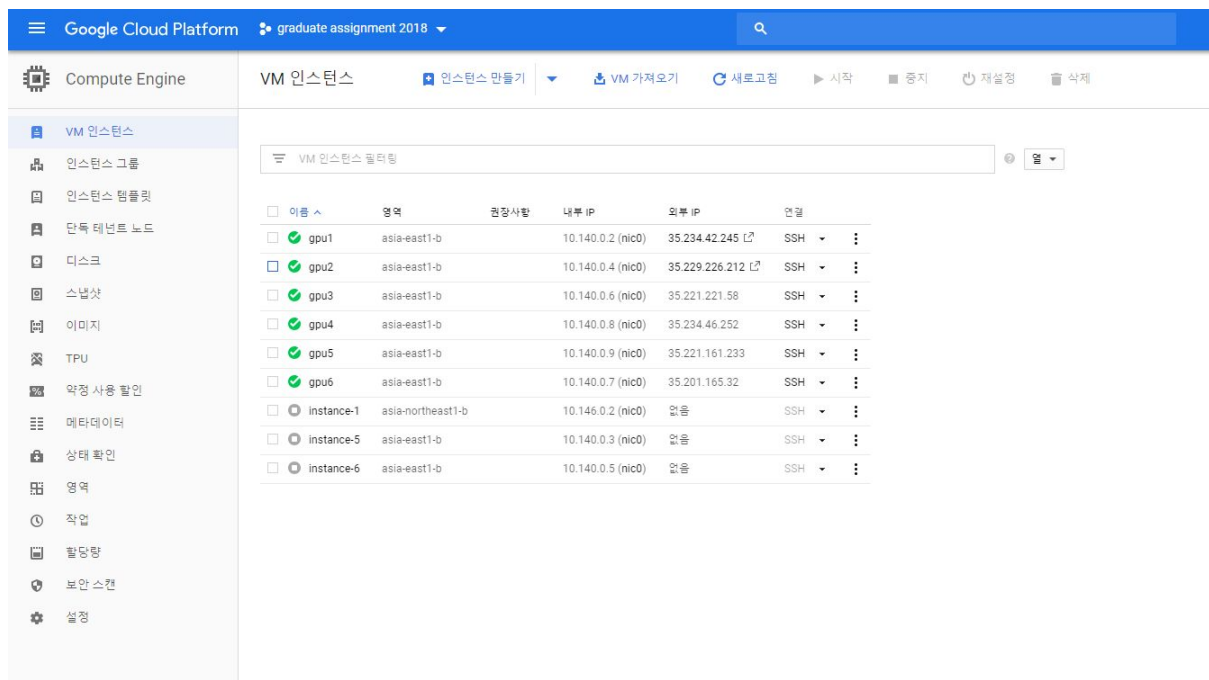
< 그림 10. 이미지 추출 >

2) 이미지 벡터 데이터 추출 방식



< 그림 11. 이미지 벡터 데이터 추출 >

5-2) 이론 학습



< 그림 12. Google Cloud Platform을 활용한 학습과정 >


```

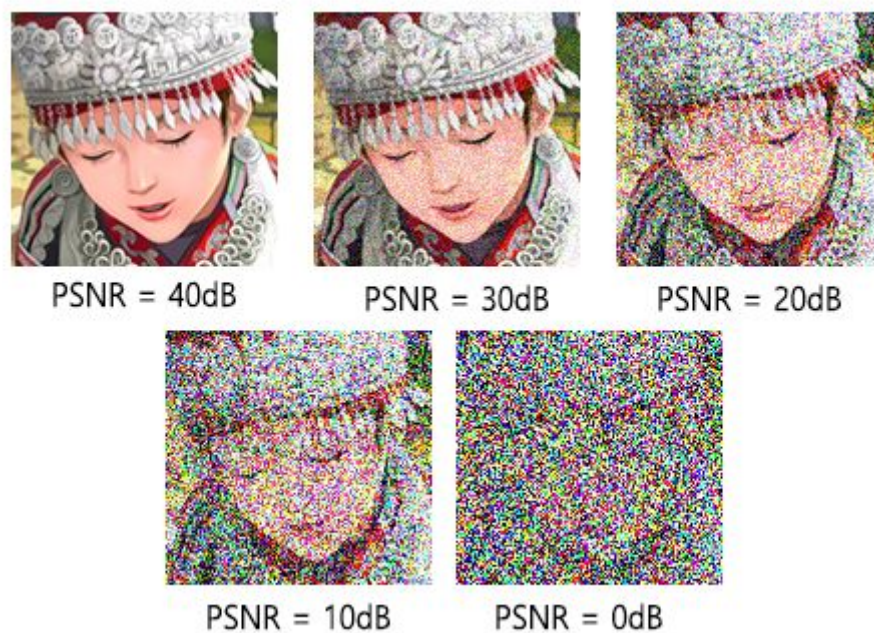
[epoch 2] loss 2720542.2500    lr 0.00010000    2714.337 sec
[epoch 3] loss 3233628.5000    lr 0.00010000    3616.416 sec
[epoch 4] loss 2951822.5000    lr 0.00010000    4518.526 sec
[epoch 5] loss 3419303.2500    lr 0.00010000    5420.835 sec
[epoch 6] loss 2889710.7500    lr 0.00010000    6323.232 sec
[epoch 7] loss 3062536.5000    lr 0.00010000    7225.435 sec
[epoch 8] loss 3415028.5000    lr 0.00010000    8128.015 sec
[epoch 9] loss 3282760.2500    lr 0.00010000    9030.389 sec
[epoch 10] loss 3180505.2500    lr 0.00001000    9932.878 sec
[epoch 11] loss 3206081.5000    lr 0.00001000    10835.923 sec
[epoch 12] loss 2609696.2500    lr 0.00001000    11738.602 sec
[epoch 13] loss 2762004.0000    lr 0.00001000    12641.018 sec
[epoch 14] loss 2273351.2500    lr 0.00001000    13543.344 sec
[epoch 15] loss 3151433.2500    lr 0.00001000    14445.708 sec
[epoch 16] loss 2729447.2500    lr 0.00001000    15347.753 sec
[epoch 17] loss 3006702.5000    lr 0.00001000    16249.564 sec
[epoch 18] loss 3291885.7500    lr 0.00001000    17151.766 sec
[epoch 19] loss 3281442.0000    lr 0.00001000    18053.949 sec
[epoch 20] loss 2393501.7500    lr 0.00000100    18956.440 sec
[epoch 21] loss 2846843.5000    lr 0.00000100    19859.143 sec
[epoch 22] loss 2867921.2500    lr 0.00000100    20761.933 sec

```

< 그림 13. Google Cloud Platform을 활용한 학습과정 >



< 그림 14. Training Set을 늘리기 위한 과정 >



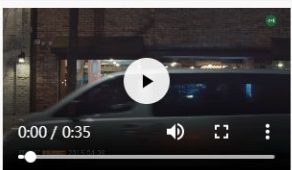
< 그림 15. 성능 평가를 위한 PSNR의 개념 >

5-3) 웹 페이지 구축

CNN을 활용한 Super Resolution

블라블라 작은 설명란

+ Add files...
Start upload
Cancel upload
Delete

Thumb	Name	Size/Progress	Button
	[MV] THE ARK(디아크) - The Light(빛) 30sec.mp4	8.98 MB	Start Cancel

Notes

- 업로드 가능한 파일 용량은 **Unlimited MB**
- 업로드 가능한 파일 확장자는 (GIF, PNG, JPG, JPEG, BMP, MP4, MKV, WMV) 만 허용합니다.
- 드래그 & 드롭으로 파일을 업로드 할 수 있습니다.
- 프로젝트가 진행되는 저장소 주소는 이며, [Documentation](#)에서 더 자세한 정보를 보실 수 있습니다.

< 그림 16. 웹 페이지 구현 >

6. 추진체계

6-3) 졸업과제 수행 단계

단계	수행 내용
1단계 (환경구축 및 조사)	<ul style="list-style-type: none"> - 인공신경망(CNN) 자료 조사 및 구축 - 사진 및 동영상 데이터 베이스 수집 - 화질 개선 알고리즘 조사
2단계 (구현)	<ul style="list-style-type: none"> - 인공신경망(CNN) 구현 - Convolutional Neural Network(CNN)를 통한 사진 및 동영상 화질 개선
3단계 (학습)	<ul style="list-style-type: none"> - 구현된 인공신경망에 특징 데이터 누적 학습
4단계 (테스트)	<ul style="list-style-type: none"> - 테스트 사이트 구축 - 최종 구현된 결과물 테스트 및 수정 보완

6-2) 구성원 역할분담

구성원	역할
김근식	학습 데이터 수집 / 화질 알고리즘 구현 및 개선
박형탁	웹 사이트 개발 / 서버&데이터베이스 구축
황남기	하이퍼 파라미터 설정&최적화 / 오버피팅 개선
공통	인공신경망을 통한 분석 학습 알고리즘

7. 추진 일정

월	5		6					7						8					9			
주차	4	5	1	2	3	4	5	1	2	3	4	5	6	1	2	3	4	5	1	2	3	4
관련지식학습																						
개발환경 구축																						
설계																						
구현 및 기계학습																						
통합 및 테스트																						
디버깅																						
결과 보고서 작성																						

8. 실험 결과 및 결론

8-1) 하이퍼 파라미터 튜닝

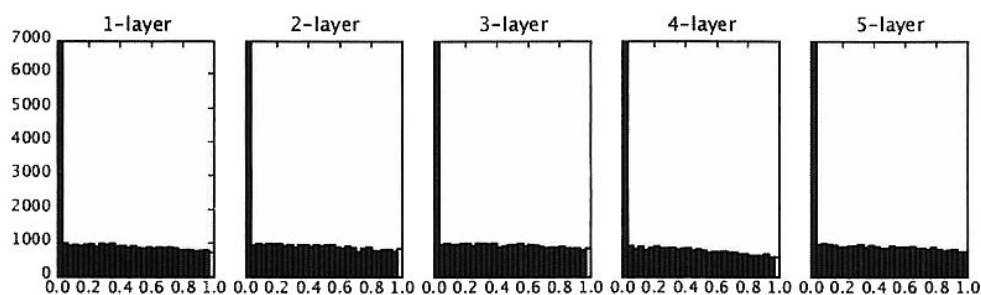
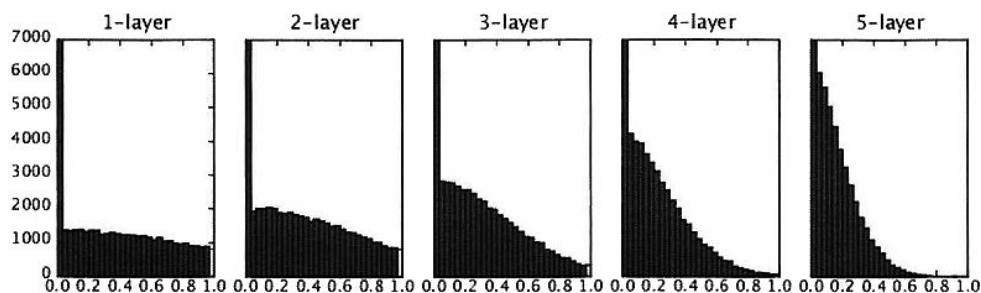
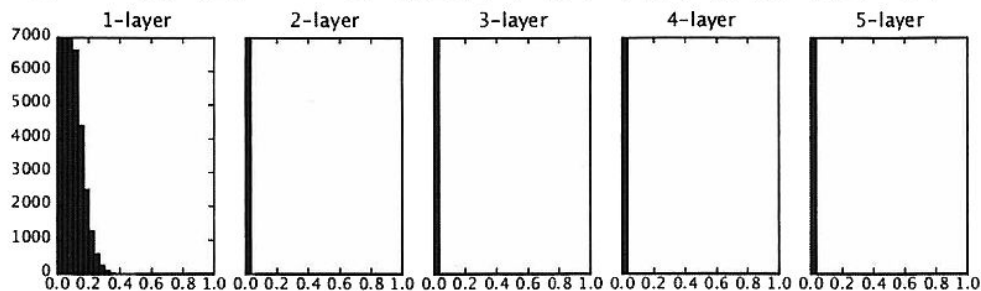
학습 관련 기술을 고려해 볼 때 다음과 같은 것들을 고려해볼 수 있다.

1. 가중치 매개변수의 최적값을 탐색하는 최적화 방법
2. 가중치 매개변수 초깃값, 하이퍼파라미터 설정 방법
3. 오버피팅 대응책 – 가중치 감소와 드롭아웃

8-1-1) SGD 이니셜라이저 비교(Xavier vs He)

먼저 Xavier 이니셜라이저와 He 이니셜라이저의 실험 결과를 비교해본다.

그림 6-14 활성화 함수로 ReLU를 사용한 경우의 가중치 초깃값에 따른 활성화값 분포 변화



< 그림17. 초깃값 설정에 따른 활성화값 분포 비교 >

결과를 보면 $\text{std}=0.01$ 일 때의 각 층의 활성화 값들은 아주 작은 값들이다. 신경망에 아주 작은 데이터가 흐른다는 것은 전파 때 가중치의 기울기 역시 작아진다는 뜻이다. 이는 중대한 문제이며, 실제로도 학습이 거의 이뤄지지 않는다.

자비에르 글로로트(Xavier Glorot)와 요슈아 벤지오(Yoshua Bengio)의 논문에서 권장하는 가중치 초기값인, 일명 Xavier 초기값을 써보았다. 이는 일반적인 딥러닝 프레임워크들이 표준적으로 이용하고 있다.

이 논문은 각 층의 활성화 값들을 광범위하게 분포시킬 목적으로 가중치의 적절한 분포를 찾고자 했는데, 앞 계층의 노드가 n 개라면 표준편차가 $1/\sqrt{n}$ 인 분포를 사용하면 된다는 결론을 이끌었다. Xavier 초기값을 사용하면 앞 층에 노드가 많을수록 대상 노드의 초기값으로 설정하는 가중치가 좁게 퍼진다.

ReLU를 이용할 때는 ReLU에 특화된 초기값을 이용하라고 권장되는데, 이 특화된 초기값을 찾아낸 카이밍 히(Kaiming He)의 이름을 따 He 초기값이라 한다. He 초기값은 앞 계층의 노드가 n 개일 때, 표준편차가 $\sqrt{2/n}$ 인 정규분포를 사용. Xavier 초기값이 $\sqrt{1/n}$ 이었다. 이는 ReLU는 음의 영역이 0이라서 더 넓게 분포시키기 위해 2배의 계수가 필요하다고 해석할 수 있다.

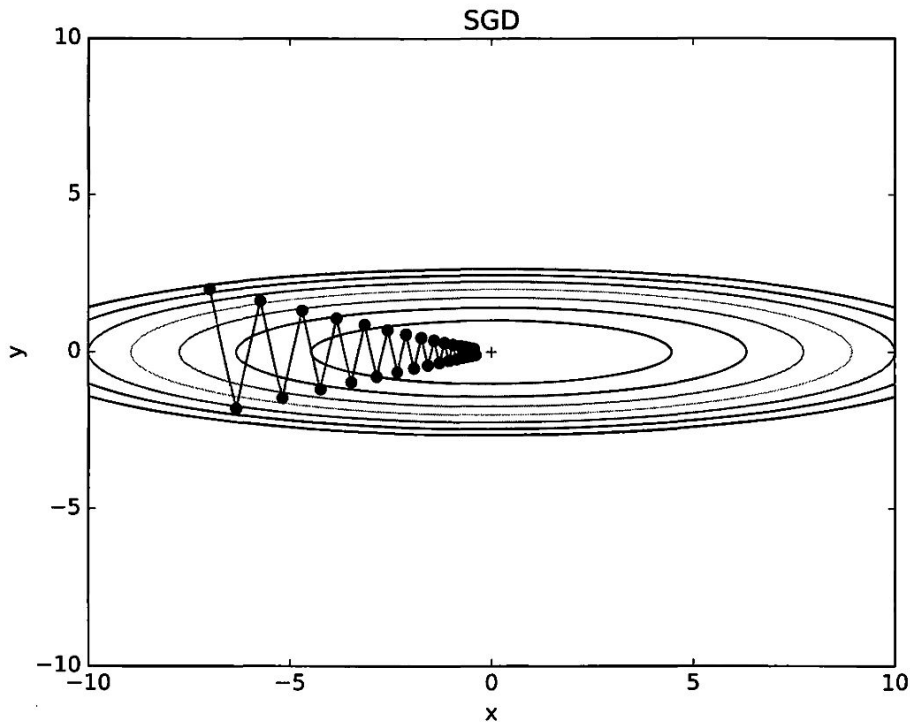
8-2) 매개변수 갱신 방법 비교(SGD, Adagrad, Momentum, Adam)

8-2-1) SGD

신경망 학습의 목적은 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것이다. 이는 곧 매개변수의 최적값을 찾는 문제인데, 이러한 문제를 푸는 것을 **최적화**(optimization)라고 한다.

최적의 매개변수 값을 찾는 단서로 매개변수의 기울기(미분)를 이용하는 방법이 **확률적 경사 하강법**(SGD)이다. 기울어진 방향으로 매개변수 값을 갱신하는 일을 몇 번이고 반복해서 점점 최적의 값에 다가가게 된다.

그림 6-3 SGD에 의한 최적화 갱신 경로 : 최솟값인 (0, 0)까지 지그재그로 이동하니 비효율적이다.



< 그림18. SGD에 의한 최적화 갱신 경로 >

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}} \quad [\text{식 6.1}]$$

< 그림19. SGD에 의한 가중치 갱신 수식 >

8-2-2) 모멘텀

모멘텀(Momentum)은 ‘운동량’을 뜻하는 단어로, 물리와 관계가 있다. 모멘텀 기법은 수식으로는 다음과 같이 쓸 수 있다.

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}} \quad [\text{식 6.3}]$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v} \quad [\text{식 6.4}]$$

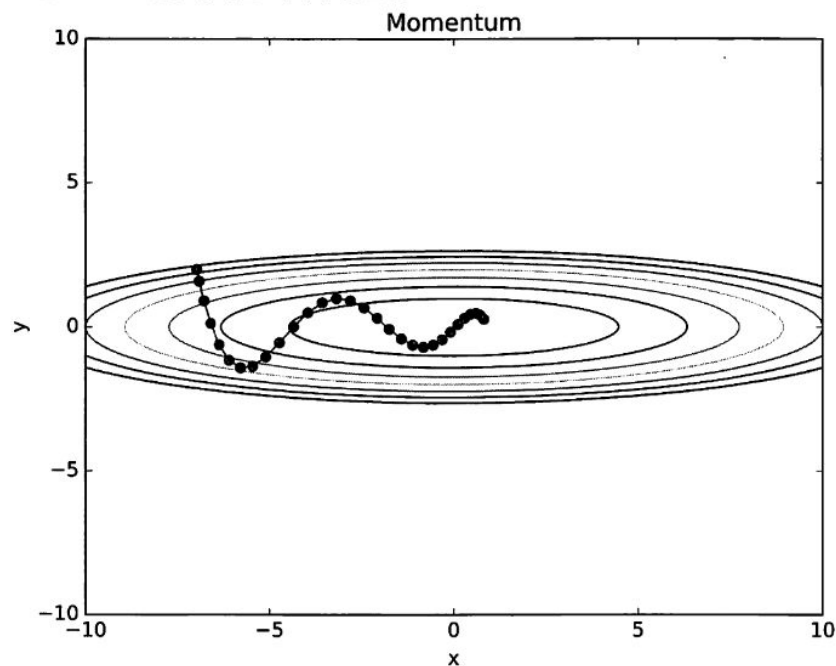
< 그림20. Momentum에 의한 가중치 갱신 수식 >

그림 6-4 모멘텀의 이미지 : 공이 그릇의 곡면(기울기)을 따라 구르듯 움직인다.

< 그림21. Momentum의 직관적 그림 >

SGD처럼 여기에서도 W 는 갱신할 가중치 매개변수, L / W 을 편미분 하는 것은 W 에 대한 손실 함수의 기울기, n 은 학습률이다. v 라는 변수가 새로 나오는데, 이는 물리에서 말하는 속도(velocity)에 해당. <그림>의 (식 6.3)은 기울기 방향으로 힘을 받아 물체가 가속된다는 물리 법칙을 나타낸다. 모멘텀은 (그림 6-4)와 같이 공이 그릇의 바닥을 구르는 듯한 움직임을 보여준다.

αv 항은 물체가 아무런 힘을 받지 않을 때 서서히 하강시키는 역할.(α 는 0.9 등의 값으로 설정) 물리에서의 지면 마찰이나 공기 저항에 해당한다.

그림 6-5 모멘텀에 의한 최적화 갱신 경로

< 그림22. Momentum에 의한 최적화 갱신 경로>

모멘텀의 갱신 경로는 공이 그릇 바닥을 굴러가듯 움직인다. SGD와 비교하면 ‘지그재그 정도’가 덜한 것을 알 수 있다. 이는 x 축의 힘은 아주 작지만 방향은 변하지 않아서 한 방향으로 일정하게 가속하기 때문이다. 거꾸로 y 축의 힘은 크지만 위아래로 번갈아 받아서 상충하여 y 축 방향의 속도는 안정적이지 않다. 전체적으로는 SGD보다 x 축 방향으로 빠르게 다가가 지그재그 움직임이 줄어든다.

8-2-3) AdaGrad

신경망 학습에서는 학습률(수식에서는 η 으로 표기) 값이 중요하다. 이 값이 너무 작으면 학습 시간이 너무 길어지고, 반대로 너무 크면 발산하여 학습이 제대로 이뤄지지 않기 때문이다.

이 학습률을 정하는 효과적 기술로 **학습률 감소**(learning rate decay)가 있다. 이는 학습을 진행하면서 학습률을 점차 줄여가는 방법이다. 처음에는 크게 학습하다가 조금씩 작게 학습한다는 얘기로, 신경망 학습에 자주 쓰인다. 학습률을 서서히 낮추는 가장 간단한 방법은 매개변수 '전체'의 학습률 값을 일괄적으로 낮추는 것이다. 이를 더욱 발전시킨 것이 AdaGrad이다. AdaGrad는 '각각의' 매개변수에 '맞춤형' 값을 만들어준다.

AdaGrad는 개별 매개변수에 적응적으로(adaptive) 학습률을 조정하면서 학습을 진행한다. AdaGrad의 갱신 방법은 수식으로는 다음과 같다.

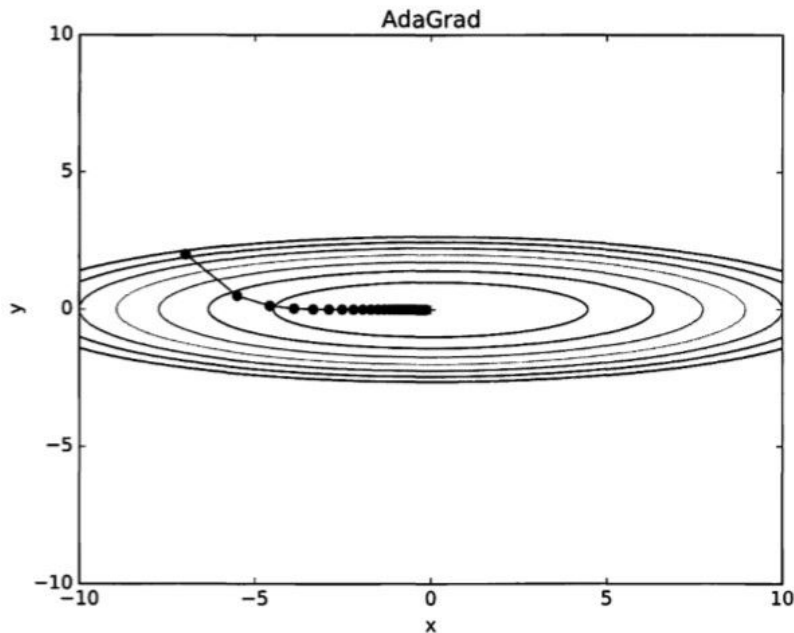
$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}} \quad [\text{식 6.5}]$$

$$\mathbf{W} \leftarrow \mathbf{W} + \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}} \quad [\text{식 6.6}]$$

< 그림23. AdaGrad에 의한 가중치 갱신 수식 >

마찬가지로 \mathbf{W} 는 갱신할 가중치 매개변수, L / \mathbf{W} 의 편미분은 \mathbf{W} 에 대한 손실 함수의 기울기, η 은 학습률. 여기에서 새로 \mathbf{h} 라는 변수가 등장한다. \mathbf{h} 는 [식 6.5]에서 보듯 기존 기울기 값을 제곱하여 계속 더해준다.(식 6.5의 동그라미 기호는 행렬의 원소별 곱셈을 의미). 그리고 매개변수를 갱신할 때 $1/\sqrt{\mathbf{h}}$ 을 곱해 학습률을 조정한다. 매개변수의 원소 중에서 많이 움직인(크게 갱신된) 원소는 학습률이 낮아진다는 뜻인데, 다시 말해 학습률 감소가 매개변수의 원소마다 다르게 적용됨을 뜻한다.

그림 6-6 AdaGrad에 의한 최적화 갱신 경로



< 그림24. AdaGrad에 의한 가중치 갱신 수식>

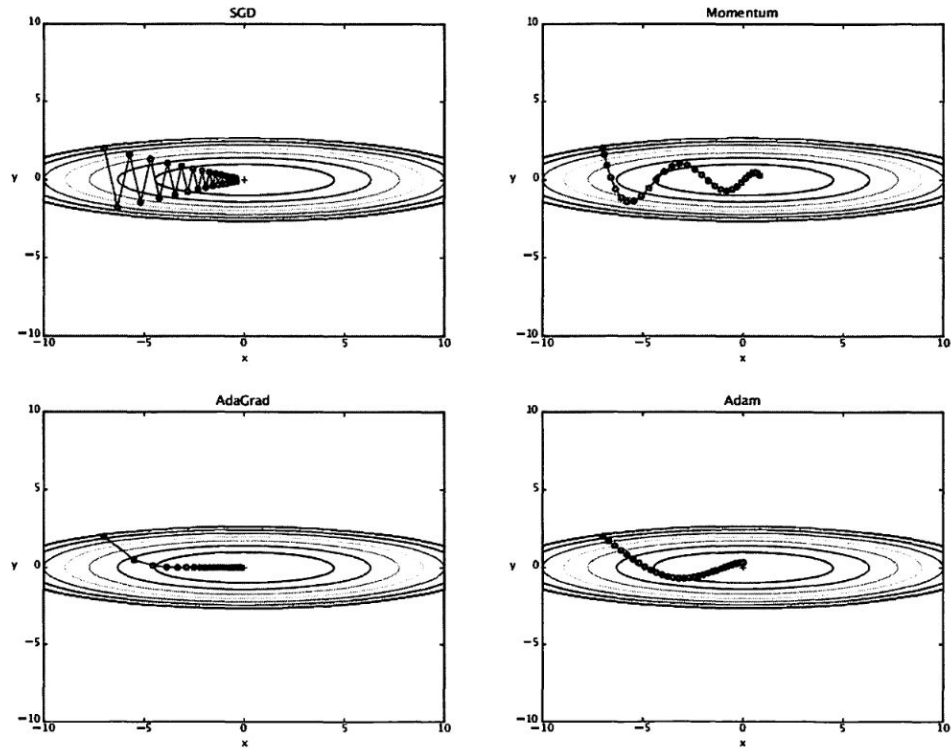
최솟값을 향해 효율적으로 움직이는 것을 알 수 있다. Y축 방향은 기울기가 커서 처음에는 크게 움직이지만, 그 큰 움직임에 비례해 갱신 정도도 큰 폭으로 작아지도록 조정된다. 그래서 y축 방향으로 갱신 강도가 빠르게 약해지고, 지그재그 움직임이 줄어든다.

8-2-4) Adam

모멘텀은 공이 그릇 바닥을 구르는 듯한 움직임을 보였다. AdaGrad는 매개변수의 원소마다 적응적으로 갱신 정도를 조정했다. “그럼 혹시 이 두 기법을 융합하면 어떻게 될까?” 라는 생각에서 출발한 기법이 바로 Adam이다. Adam은 2015년에 제안된 새로운 방법이다. 이론은 다소 복잡한데, 직관적으로는 모멘텀과 AdaGrad를 융합한 듯한 방법이라 볼 수 있다. 하이퍼파라미터의 ‘편향 보정’이 진행된다는 점도 Adam의 특징이다.

8-2-5) 최적화 기법들 비교

그림 6-8 최적화 기법 비교 : SGD, 모멘텀, AdaGrad, Adam

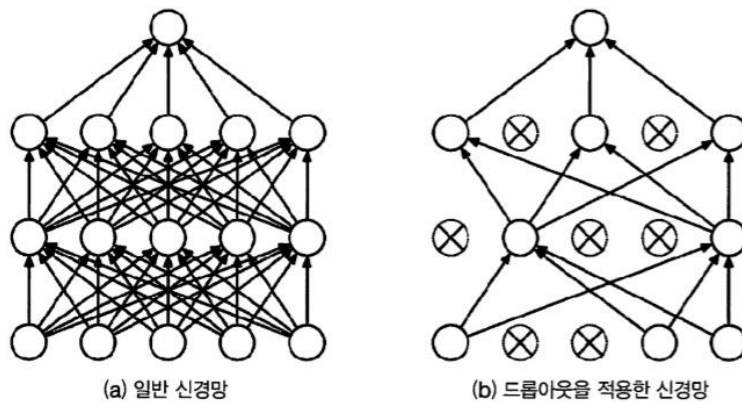


< 그림25. 최적화 기법 비교 >

8-3) 오버피팅 개선 위한 드롭아웃과 배치학습 방법

드롭아웃

그림 6-22 드롭아웃의 개념(문헌¹⁴⁾에서 인용) : 왼쪽이 일반적인 신경망, 오른쪽이 드롭아웃을 적용한 신경망. 드롭아웃은 뉴런을 무작위로 선택해 삭제하여 신호 전달을 차단한다.

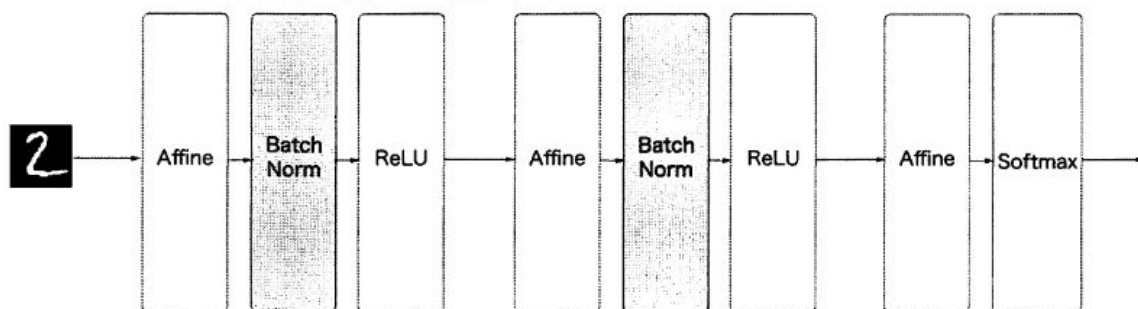


< 그림26. 드롭아웃 적용 유무 비교 >

드롭아웃은 뉴런을 임의로 삭제하면서 학습하는 방법이다. 훈련 때 은닉 층의 뉴런을 무작위로 골라 삭제한다. 삭제된 뉴런은 <그림-26> 과 같이 신호를 전달하지 않게 된다. 훈련 때는 데이터를 흘릴 때마다 삭제할 뉴런을 무작위로 선택하고, 시험 때는 모든 뉴런에 신호를 전달한다. 단, 시험 때는 각 뉴런의 출력에 훈련 때 삭제한 비율을 곱하여 출력한다.

배치학습

그림 6-16 배치 정규화를 사용한 신경망의 예



< 그림27. 배치 정규화를 사용한 신경망 >

배치 정규화는 2015년에 제안된 방법이다. 배치 정규화는 아직 세상에 나온 지 얼마되지 않은 기법임에도 많은 연구자와 기술자가 즐겨 사용하고 있다. 실제로 기계학습 콘테스트의 결과를 보면 이 배치 정규화를 사용하여 뛰어난 결과를 달성한 예가 많다.

배치 정규화가 주목받는 이유는 다음과 같다.

- 학습을 빨리 진행할 수 있다(학습 속도 개선)
- 초기값에 크게 의존하지 않는다(골치 아픈 초기값 선택 필요 없음).
- 오버피팅을 억제한다(드롭아웃 등의 필요성 감소).

딥러닝의 학습 시간이 길다는 걸 생각하면 첫 번째 이점은 아주 반가운 일이다. 초기값에 크게 신경 쓸 필요가 없고, 오버피팅 억제 효과가 있다는 점도 딥러닝 학습의 걱정거리를 덜어준다.

배치 정규화의 기본 아이디어는 각 층에서의 활성화값이 적당히 분포되도록 조정하는 것이다. 그래서 [그림 27]과 같이 데이터 분포를 정규화하는 '배치 정규화 Batch Norm 계층'을 신경망에 삽입한다.

배치 정규화는 그 이름과 같이 학습 시 미니배치를 단위로 정규화한다. 구체적으로는 데이터 분포가 평균이 0, 분산이 1이 되도록 정규화한다. 수식은 다음과 같다.

$$\begin{aligned}\mu_B &\leftarrow \frac{1}{m} \sum_{i=1}^m x_i \\ \sigma_B^2 &\leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \\ \hat{x}_i &\leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\end{aligned}\quad \text{[식 6.7]}$$

< 그림 28. 배치 정규화 수식 >

여기에는 미니배치 $B = \{x_1, x_2, \dots, x_m\}$ 이라는 m 개의 입력 데이터의 집합에 대해 평균 μ_B 와 분산 σ_B^2 을 구한다. 그리고 입력 데이터를 평균이 0, 분산이 1이 되게(적절한 분포가 되게) 정규화한다. <그림 28>에서 기호 입실론은 작은 값(예컨대 $10e-7$ 등)으로, 0으로 나누는 사태를 예방하는 역할이다.

<그림 28>은 단순히 미니배치 입력 데이터 $\{x_1, x_2, \dots, x_m\}$ 을 평균 0, 분산 1인 데이터 $\{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m\}$ 으로 변환하는 일을 한다. 이 처리를 활성화 함수의 앞(혹은 뒤)에 삽입함으로써 데이터 분포가 덜 치우치게 할 수 있다.

또 배치 정규화 계층마다 이 정규화된 데이터에 고유한 확대와 이동 변환을 수행한다. 수식으로는 다음과 같다.

$$y_i \leftarrow \gamma \hat{x}_i + \beta \quad \text{[식 6.8]}$$

< 그림 29. 데이터의 고유한 확대와 이동 변환 수식 >

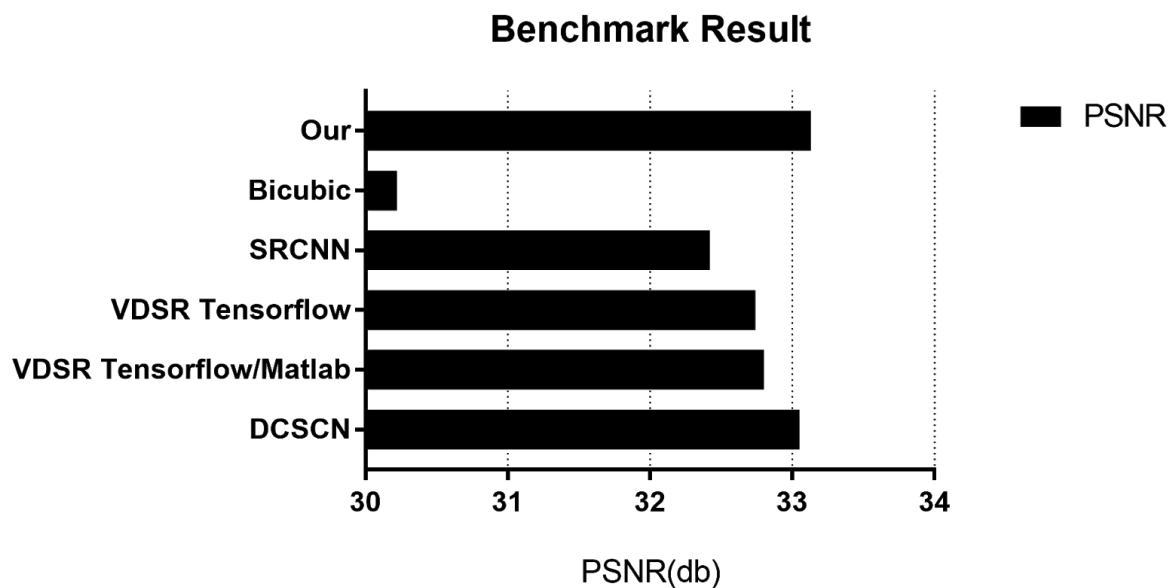
이 식에서 Gamma가 확대를, 베타가 이동을 담당한다. 두 값은 처음에는 Gamma = 1, Beta = 0부터 시작하고, 학습하면서 적합한 값으로 조정해간다. 이상이 배치 정규화의 알고리즘이다.

8-2) 실험 결과 및 결론

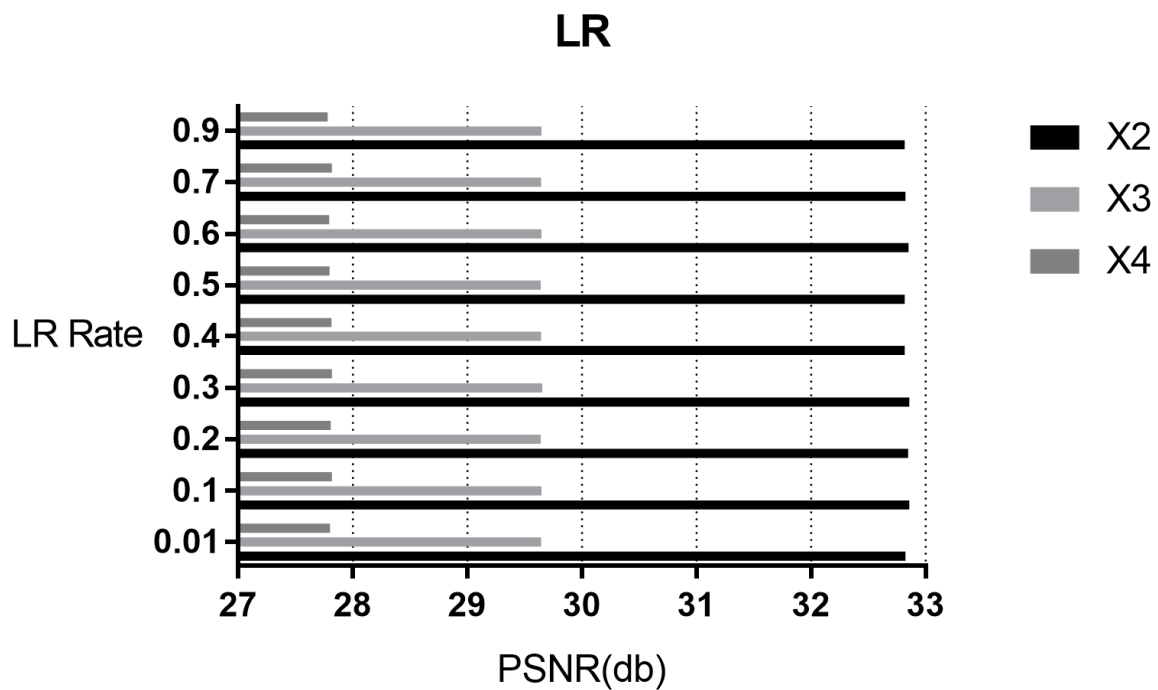
이번 과제로 인해 콘텐츠 제작비 절감으로 더욱 양질의 콘텐츠가 제작 가능해지고, Platform화로 인해 일반인도 손쉽게 사용할 수 있게 되어, 고해상도 콘텐츠에 대한 접근성이 강화될 것으로 기대된다. 나아가, 저해상도-고 잡음 영상 복원으로 범인 인상착의 파악 등 다양한 분야에서 폭넓은 활용이 가능해질 것으로 예상된다.



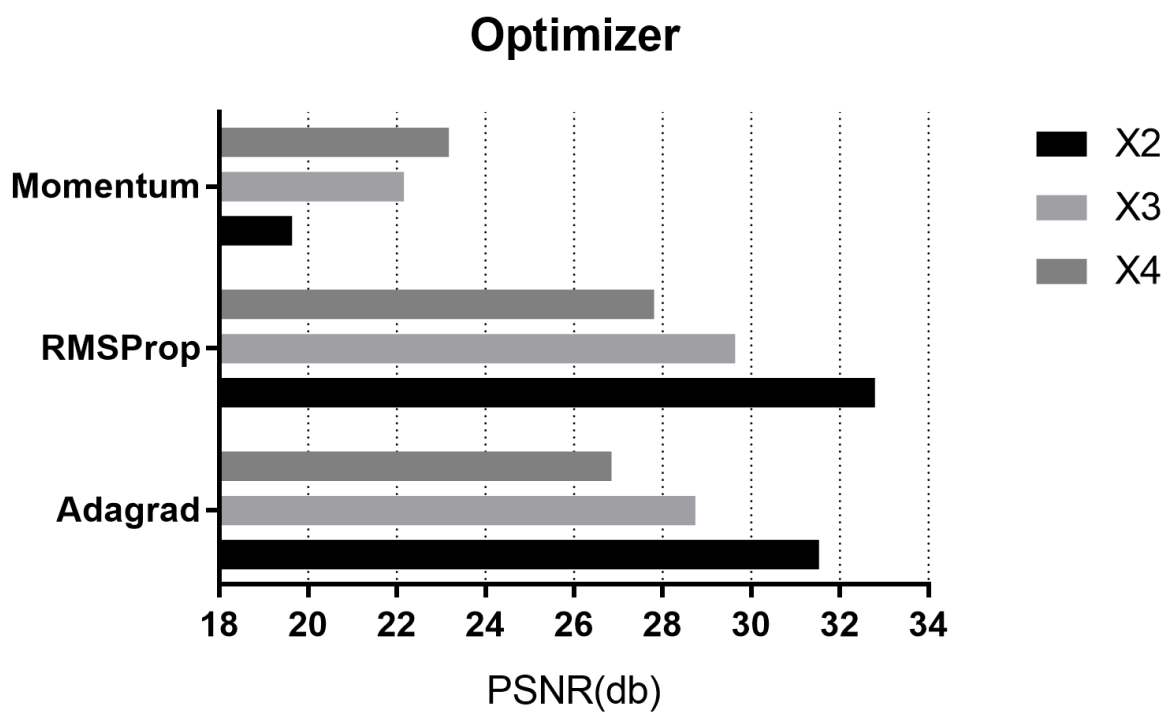
< 그림30. 플랫폼을 거친 화질개선 결과 >



< 그림31. 다양한 방법들에 의한 PSNR 비교 >



< 그림32 학습률에 따른 PSNR 결과 >



< 그림33. Optimizer별 PSNR 결과 >

9. 참고 레퍼런스

- Accurate Image Super-Resolution Using Very Deep Convolutional Networks, Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee (https://cv.snu.ac.kr/research/VDSR/VDSR_CVPR2016.pdf)
- Image Super-Resolution Using Deep Convolutional Networks, Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang (<http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html>)
- Accelerating the Super-Resolution Convolutional Neural Network, Chao Dong, Chen Change Loy, Xiaoou Tang (<http://mmlab.ie.cuhk.edu.hk/projects/FSRCNN.html>)
- 밑바닥부터 시작하는 딥러닝 - 사이토 고키, 한빛 미디어 2017.
- 신경망 첫걸음 - 타리크 라시드, 한빛 미디어, 2017.