



딥러닝 기반 COVID-19 Low Dose CT 영상 복원

허지혜 이수빈 정예지
민지원 임현영 정보경 하지원



CONTENTS

01 소개

- 01. 팀 소개
- 02. 연구 진행
- 02. 연구 주제

02 데이터 설명

- 01. 데이터 설명
- 02. 영상 전처리 과정

03 모델링

- 01. Deep Learning
- 02. SRCNN
- 03. FSRCNN
- 04. ESPCN

04 결론

- 01. 모델 결과
- 02. 결론 및 아쉬운 점
- 03. 출처

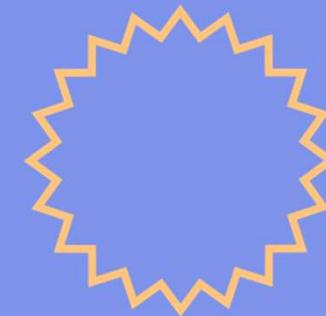
01



01. 팀 소개

02. 연구 진행

03. 연구 주제



01. 팀 소개

대학원생 연구 책임자, 대학생 연구 팀원



허지혜(연구책임자)

경상국립대학교
정보통계학과 석사 과정
전공 : 빅데이터



이수빈(연구 팀원)

경상국립대학교
수학과 재학
역할 : 세미나 진행 및 동영상 제작

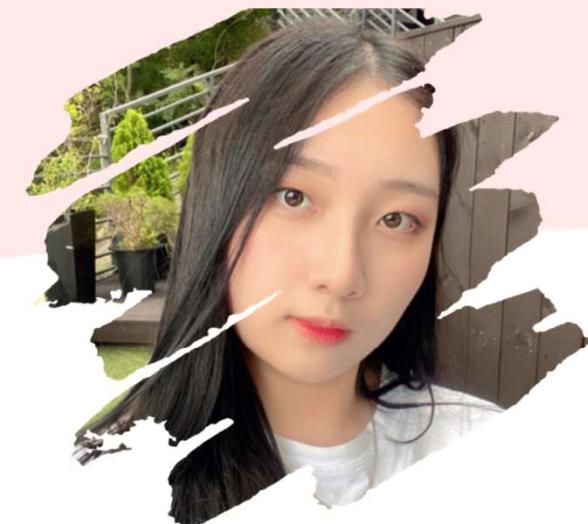


정예지(연구 팀원)

경상국립대학교
수학과 재학
역할 : 세미나 진행 및 보고서 제출

01. 팀 소개

고등학생 연구 팀원



민지원(연구 팀원)



정보경(연구 팀원)



임현영(연구 팀원)



하지원(연구 팀원)

상문고등학교
역할 : 세미나 진행, 숙제,보고서 제출

물금고등학교
역할 : 세미나 진행, 보고서 제출 및 발표

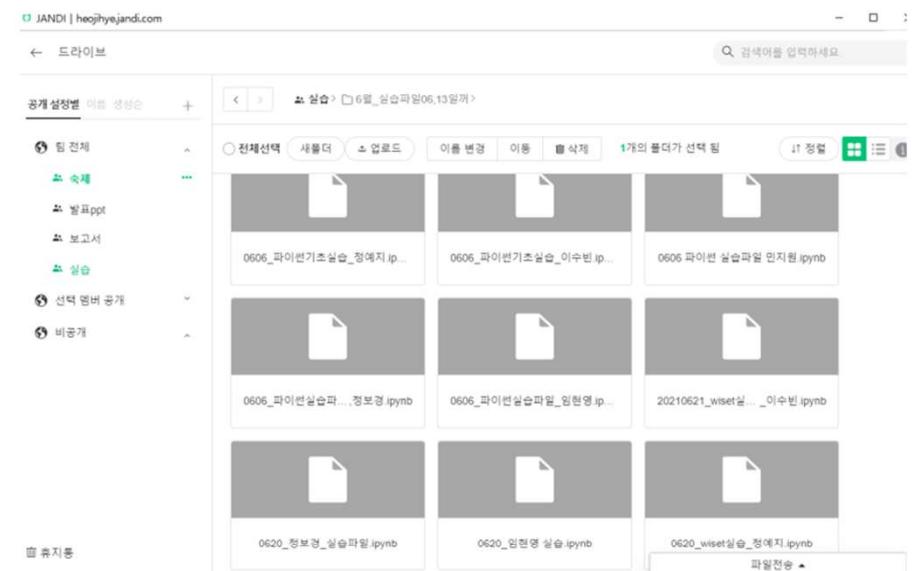
물금고등학교
역할 : 세미나 진행, 숙제,보고서 제출

상문고등학교
역할 : 세미나 진행,숙제,보고서 제출

02. 연구 진행

역할 분담 - 단체 세미나

월	내용	허지혜	이수빈	정예지	고등학생
5월	파이썬 환경 설치	SUB	MAIN	SUB	
5월	파이썬 기본문법1	SUB			MAIN
6월	파이썬 기본문법2	SUB			MAIN
6월	CT 촬영 구조 및 영상 불러오기 방법	MAIN			
7월	산업체 멘토 미팅				
7월	딥러닝 개념	SUB		MAIN	
8월	파이토치 기본문법	SUB	MAIN		
8월	이미지 처리 전문가 미팅				
9월	딥러닝 모델 파이토치 구현	SUB		MAIN	
9월	논문 담당 전문가 미팅				
10월	최종 모형 설명	MAIN			
10월	최종 모델 발표	MAIN		MAIN	MAIN
	숙제	MAIN			



<잔디 앱으로 숙제 제출>

02. 연구 진행

역할 분담 - 단체 세미나

정예지님이 발표 중입니다.

Python 기초

Python: 인터프리터 언어

인터프리터	컴파일러
원시코드를 기계어로 변환 과정 없이 한줄씩 해석 후 실행	작성한 코드를 한번에 기계어로 번역
바이너리 파일 없음	번역 후 코드를 하나의 바이너리 파일로 저장
에러발생 후 코드 실행이 안됨	전체 코드를 번역 후 에러보고
번역 속도 빠름	번역 속도 느림
프로그램 실행 속도 느림	프로그램 실행 속도 빠름

*컴파일러: 프로그래밍 언어를 기계어가 이해할 수 있는 언어로 변환해주는 것.

회의 세부정보 ^

자막 사용 정예지 님이 발표 중입니다.

오후 2:07

경

나 정예지 하지혜 현영 지원 지원

경

<8월 파이토치 기본문법 모습>

경 보경님이 발표 중입니다.

튜플 자료형

튜플 만드는 방법

리스트	튜플
[]으로 둘러싸여 짐	()으로 둘러싸여 짐
값의 생성, 삭제, 수정 가능	값을 바꿀 수 없다

튜플 만들기
튜플은 리스트와 같은 데이터 구조로 사용되는 자료형입니다. 주요 특징은 다음과 같습니다.
1. 리스트는 []으로 둘러싸여 짐입니다. 예제: [1, 2, 3]
2. 튜플은 ()으로 둘러싸여 짐입니다. 예제: (1, 2, 3)
3. 튜플은 리스트와 같은 형식으로 값을 추가하거나 제거하는 연산이 가능합니다.
4. 튜플은 리스트와 같은 형식으로 값을 수정하는 연산이 가능합니다.
5. 튜플은 리스트와 같은 형식으로 값을 삭제하는 연산이 가능합니다.
6. 튜플은 리스트와 같은 형식으로 값을 찾거나 정렬하는 연산이 가능합니다.
7. 튜플은 리스트와 같은 형식으로 값을 비교하는 연산이 가능합니다.
8. 튜플은 리스트와 같은 형식으로 값을 출력하는 연산이 가능합니다.
9. 튜플은 리스트와 같은 형식으로 값을 읽어들이는 연산이 가능합니다.
10. 튜플은 리스트와 같은 형식으로 값을 쓰는 연산이 가능합니다.
11. 튜플은 리스트와 같은 형식으로 값을 넣는 연산이 가능합니다.
12. 튜플은 리스트와 같은 형식으로 값을 뺀다.

경 보경 정예지 지원

경

<5월 파이썬 기본문법 모습>

02. 연구 진행

역할 분담 - 대학생 세미나

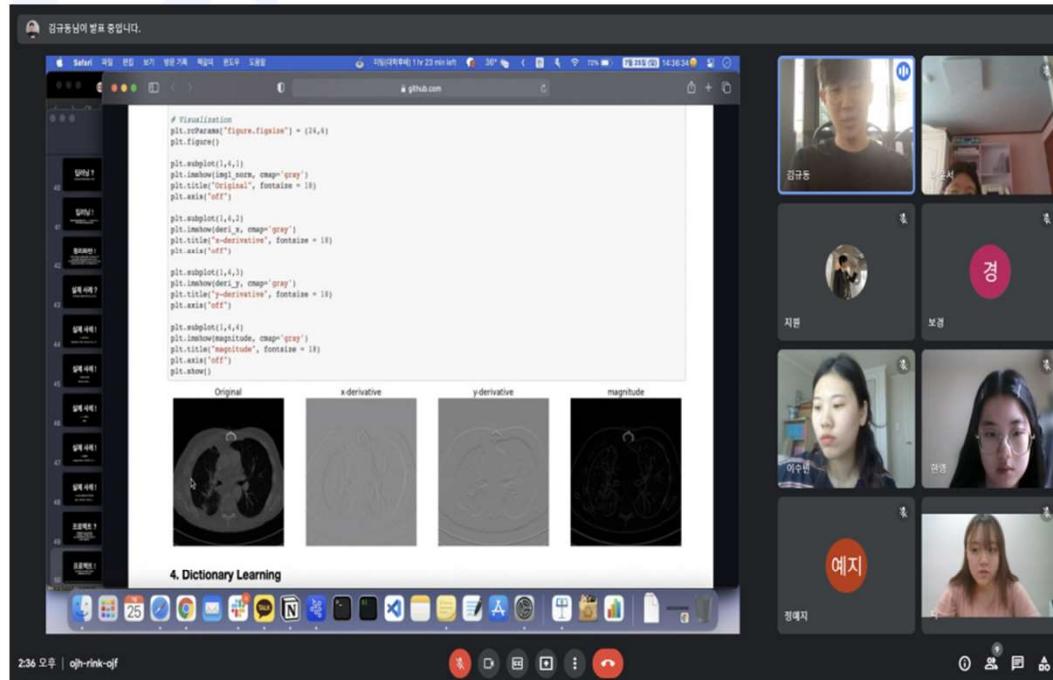
월	내용	허지혜	이수빈	정예지
4월	파이썬 기본문법	MAIN		
5월	파이썬 딥러닝 모형1	SUB	MAIN	
5월	파이썬 딥러닝 모형2	SUB		MAIN
6월	CT 촬영 구조 및 영상 불러오기 방법	MAIN		
7월	산업체 멘토 미팅			
7월	파이토치 기본문법1	SUB		MAIN
8월	딥러닝 모형 파이토치구현	SUB	MAIN	
8월	이미지 처리 전문가 미팅			
9월	딥러닝 모형 공부	SRCNN, FSRCNN	ESPCN	분류 모델링
9월	논문 담당 전문가 미팅			
9월	선행 논문 조사	MAIN		
10월	최종 논문 작성	MAIN	서론, ESPCN	서론, 결론 정리



<5월 대면 세미나 당시 모습>

02. 연구 진행

역할 분담 - 대학생 세미나



<7월 산업체 멘토 강연 모습>



<9월 전문가 활용 논문 첨삭 모습>

03. 연구 주제

COVID-19 심각성

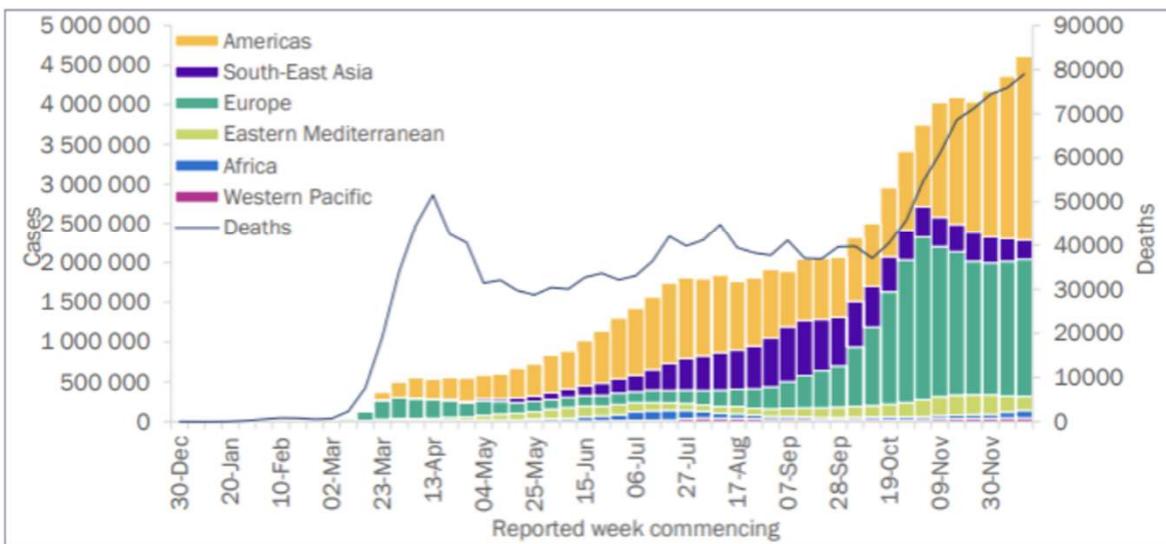


그림 1. COVID-19 확진자 주간보고(WHO 지역별 확진자, 사망자, as 20 December, 2020)

03. 연구 주제



RT-PCR 은 DNA를 증폭시키는 중합효소 연쇄반응(PCR)의 한 종류로 바이러스를 진단하는데 효과적이다.



낮은 정확도

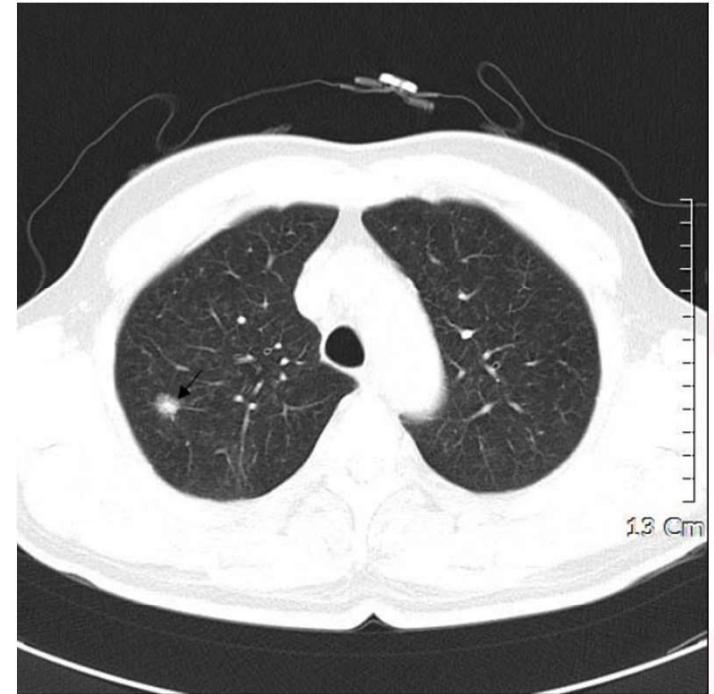
RT-PCR 검사는 검체의 종류, 검체 채취 방법 및 보관에 따라 검사 결과의 변동이 쉽게 올 수 있다. 또한 감염 후 검사 시점에도 영향을 받아 낮은 정확도를 가지고 있다.



오래 걸리는 시간

RT-PCR 검사는 검사 후 결과의 도출까지 '하루'라는 긴 시간을 가지고 있다.

03. 연구 주제



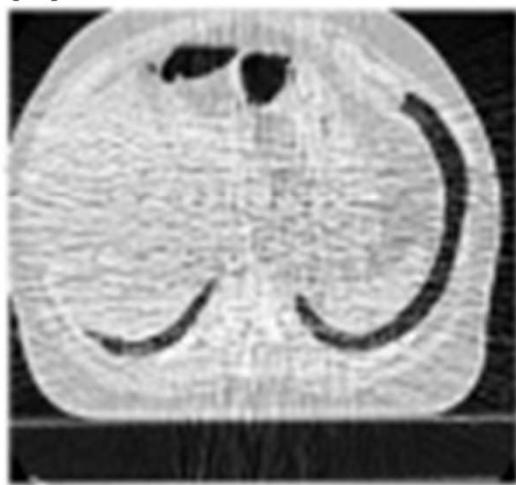
출처 : <http://www.yangsanilbo.com/news/articleView.html?idxno=42587>

03. 연구 주제

(1)



(2)



(1) 방사선 다량 투과 -> 선명한 고화질 CT 영상

(2) 방사선 소량 투과 -> 저화질 CT 영상

03. 연구 주제

저화질 CT 영상

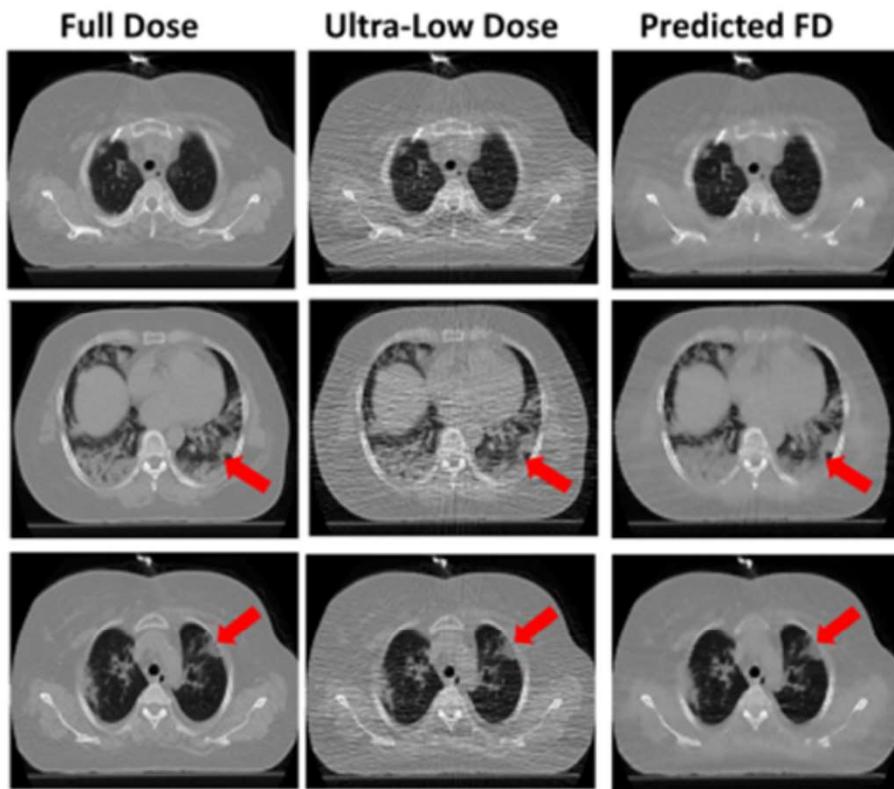


딥러닝 모형



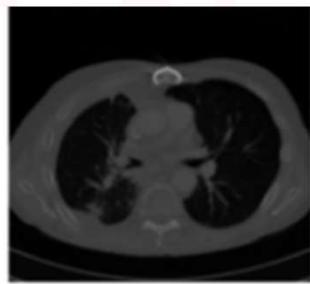
복원된 고화질 CT 영상

03. 연구 주제

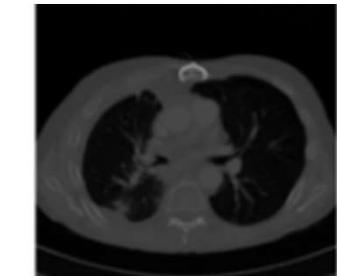


하지만, 고화질 CT 영상과 저화질 CT 영상이
같이 있는 데이터셋이 없기 때문에 임의로
생성해줘야 한다.

03. 연구 주제



CT Image



생성된 고화질 CT Image



CT Image

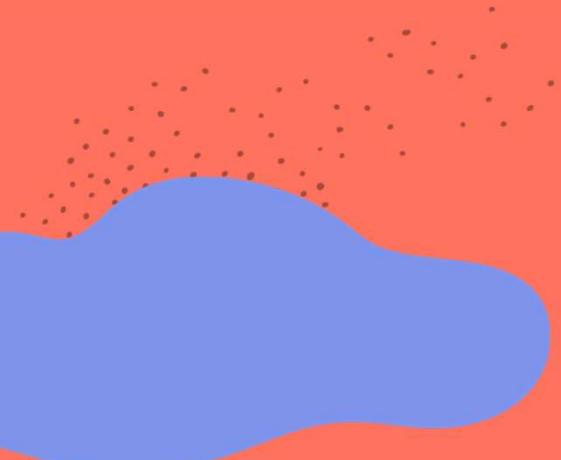
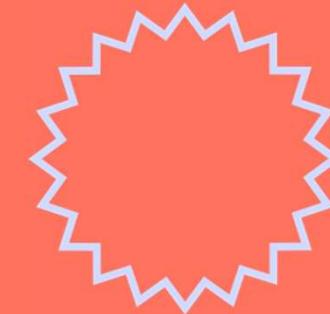
03. 연구 주제

“ Pytorch로 구현하는 딥러닝 기반 Low Dose CT 영상 복원 ”

02



- 01. 데이터 설명
- 02. 영상 전처리 과정



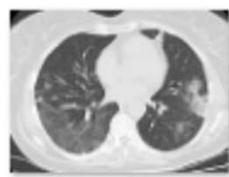
01. 데이터 설명

UCSD COVID-CT database 자료

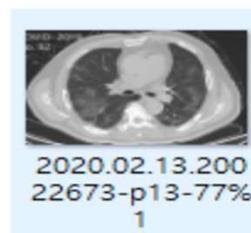
216명의 환자에서 가져온 349개의 COVID-19 CT Image



14%1



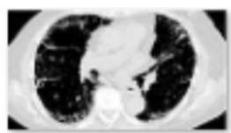
14%2



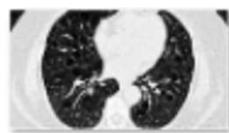
2020.02.13.200
22673-p13-77%
1



2020.02.13.200
22673-p13-77%
2



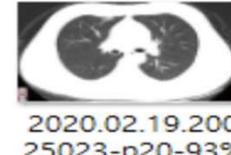
15%2



15%3



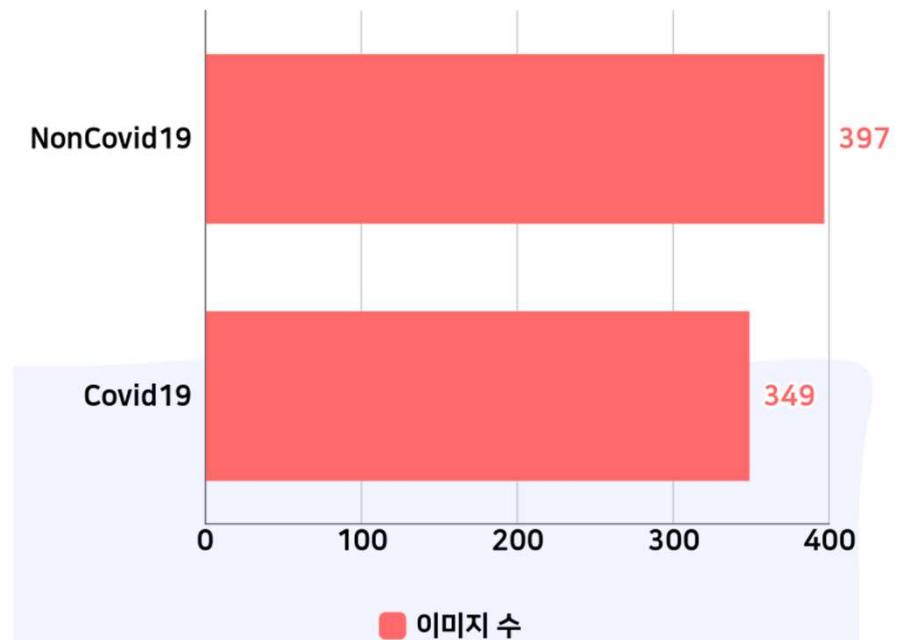
2020.02.17.200
24018-p17-61%
4



2020.02.19.200
25023-p20-93%
0

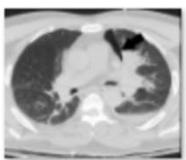
NonCovid19 Image

Covid19 Image

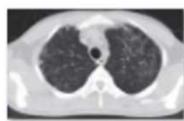


02. 영상 전처리 과정

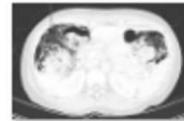
영상 데이터의 크기가 다 다르기 때문에
(299,299)로 크기를 맞춰주었다.



57%



59%



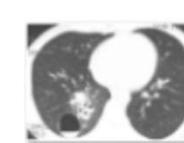
59



60



62



63%

02. 영상 전처리 과정

```
from PIL import Image  
  
for i in range(b):  
    a = os.listdir('./DATA/basic/0/')  
    img = Image.open('./DATA/basic/0/' + a[i])  
    img = img.resize((299,299))  
    img = img.convert('RGB')  
    form1 = '{}.jpg'.format(i)  
    img.save(data_path + form1)  
    #img.show()
```

영상 데이터의 크기가 다 다르기 때문에
(299,299)로 크기를 맞춰주었다.

영상의 컬러가 날아가는 것을 방지하기 위해
RGB 컬러로 한번 더 정의해주었다.

영상 저장 확장자가 달라 JPG로 만들어주었다.

02. 영상 전처리 과정

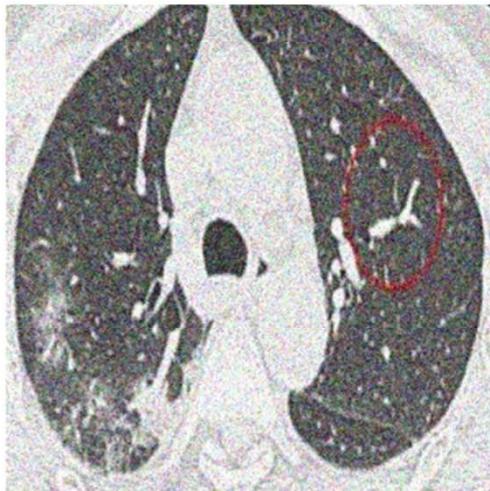
일반 CT 영상에서 저화질 CT 영상으로 변환하기 위해 Noise를 추가하는 작업

1) Gaussian Noise

정규분포를 가지는 잡음. 어느 정도 랜덤하면서 쉽게 볼 수 있는 일반적인 잡음을 의미한다.



일반 CT 영상



Gauss Noise를 추가한 CT 영상

```
if noise_typ == "gauss":  
    row,col,ch= image.shape  
    mean = 0  
    var = 1  
    sigma = var**0.5  
    gauss = np.random.normal(mean,sigma,(row,col,ch))  
    gauss = gauss.reshape(row,col,ch)  
    noisy = image + gauss  
return noisy
```

gauss 분포 생성



gauss 분포의 shape를 image처럼 변경



기존 image에 gauss 잡음 추가

02. 영상 전처리 과정

일반 CT 영상에서 저화질 CT 영상으로 변환 하기 위해 Noise를 추가하는 작업

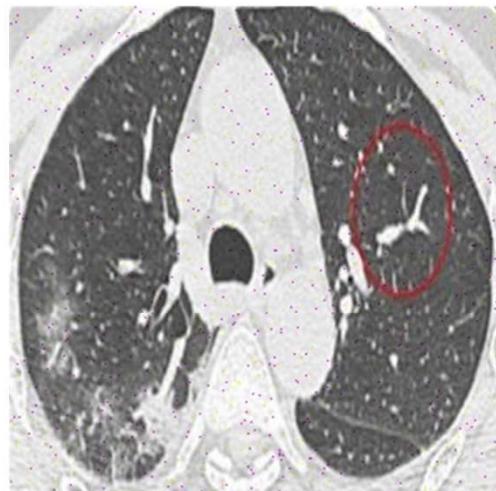
2) S&P Noise

소금&후추 잡음을 의미하며 흰색과 검정색 잡음으로 이루어진다.

즉, s&p 잡음은 입력 영상의 임의의 좌표 픽셀 값을 0 또는 255로 만드는 형태의 잡음이다.



일반 CT 영상



s&p Noise를 추가한 CT 영상

```
elif noise_typ == "s&p":  
    row,col,ch = image.shape  
    s_vs_p = 0.5  
    amount = 0.004  
    out = np.copy(image)  
    # Salt mode  
    num_salt = np.ceil(amount * image.size * s_vs_p)  
    coords = [np.random.randint(0, i - 1, int(num_salt))  
              for i in image.shape]  
    out[coords] = 1  
  
    # Pepper mode  
    num_pepper = np.ceil(amount* image.size * (1. - s_vs_p))  
    coords = [np.random.randint(0, i - 1, int(num_pepper))  
              for i in image.shape]  
    out[coords] = 0  
return out
```

02. 영상 전처리 과정

일반 CT 영상에서 저화질 CT 영상으로 변환 하기 위해 Noise를 추가하는 작업

3) Poisson Noise

평균이 Lamda인 포아송 분포를 따르는 확률 변수를 잡음 정도와 곱한 정도의 잡음이다.



일반 CT 영상



Poisson Noise를 추가한 CT 영상

```
elif noise_typ == "poisson":  
    vals = len(np.unique(image))  
    vals = 2 ** np.ceil(np.log2(vals))  
    noisy = np.random.poisson(image * vals) / float(vals)  
    return noisy
```

image의 유일한 값의 길이를 구해 vals 변수에 넣기



vals를 log 치환을 하여 소수점을 잘라낸다.

이를 지수로, 2를 밑으로 정의한다.

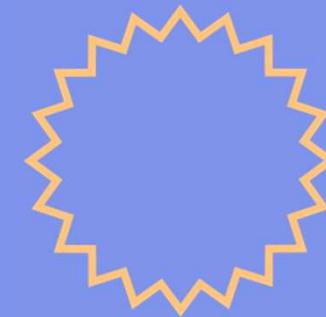


image 값에 vals를 곱해 vals의 길이로 나눈 값 만큼의
poisson 분포를 따르게 한다.

03

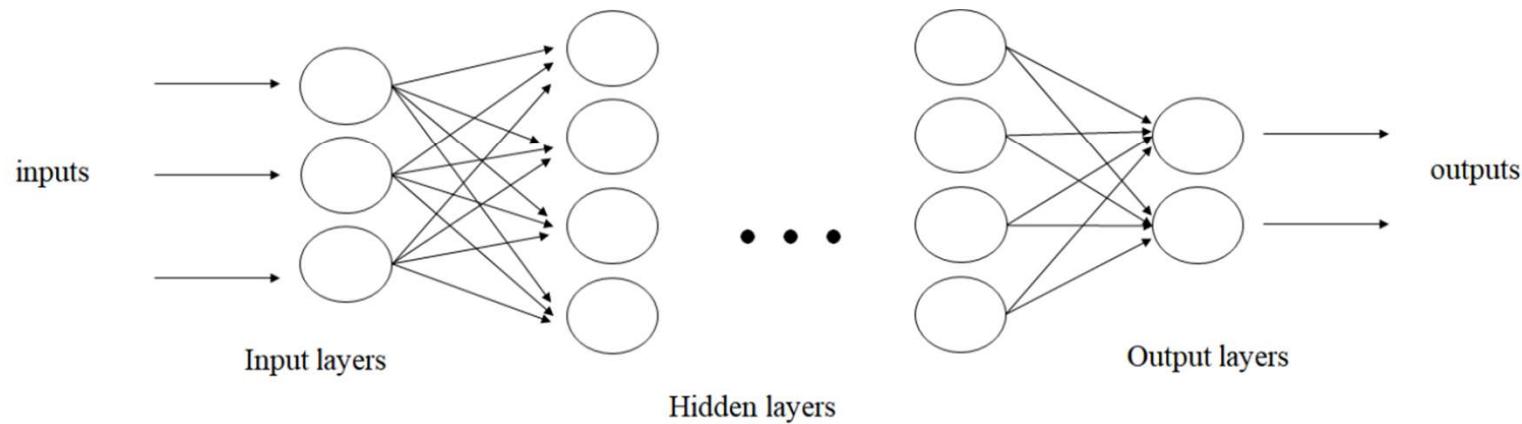


- 01. Deep Learning
- 02. SRCNN
- 03. FSRCNN
- 04. ESPCN



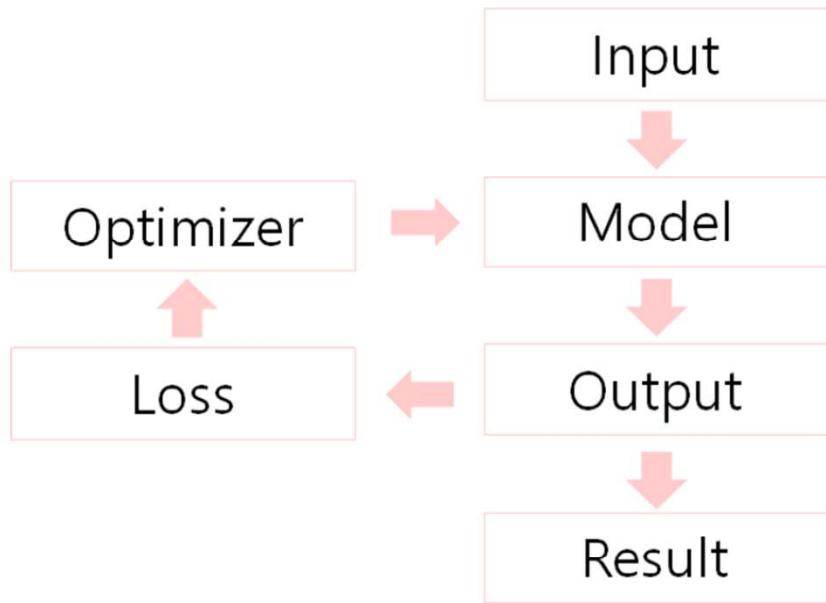
01. Deep Learning

딥러닝이란 다중 구조의 신경망을 기반으로 한 머신러닝의 한 분야이다.



01. Deep Learning

딥러닝 process는 다음과 같다.



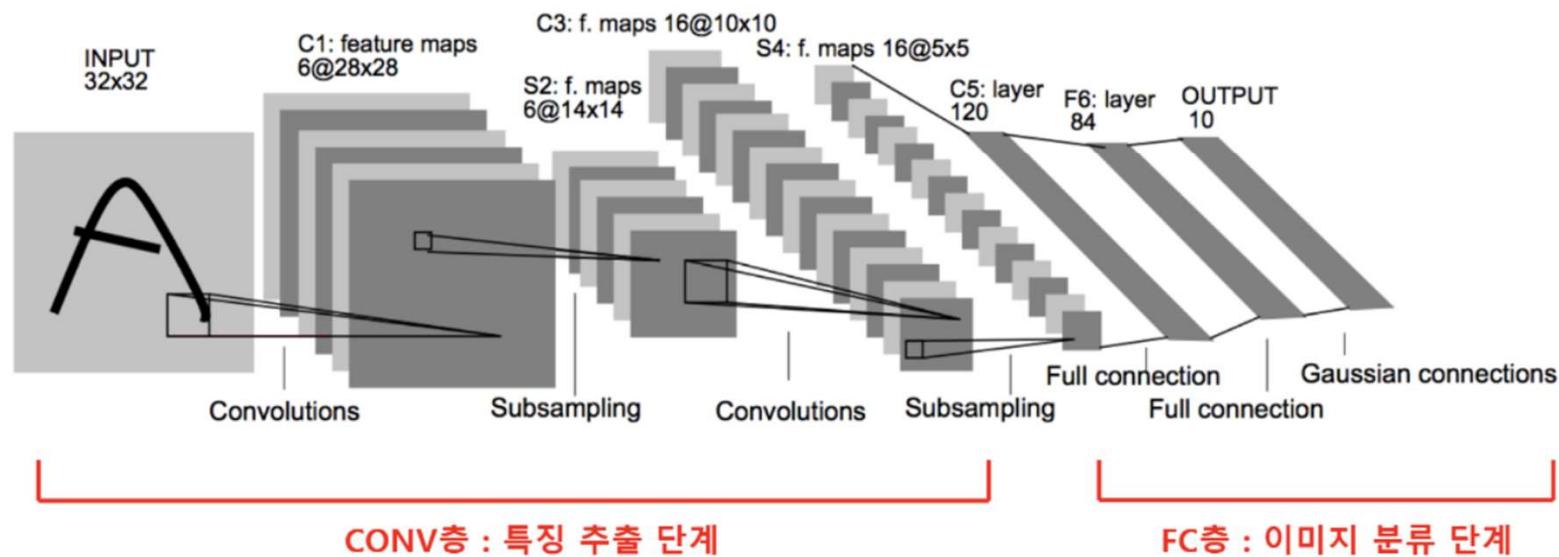
딥러닝에 몇가지 주요 개념을 이야기 하면 다음과 같다.

- 1) **loss 함수** : 학습을 통해 생성된 모델의 예측값과 실제값 간의 차이를 나타내는 함수이다.
- 2) **optimizer** : 가중치와 편향을 갱신하는 방법이다. 이때 예측값과 실제값의 차이를 최소화하기 위해 파라미터를 수정하는 방향으로 간다.
- 3) **learning rate** : 얼마만큼의 스텝으로 전진할지를 결정하는 파라미터이다.

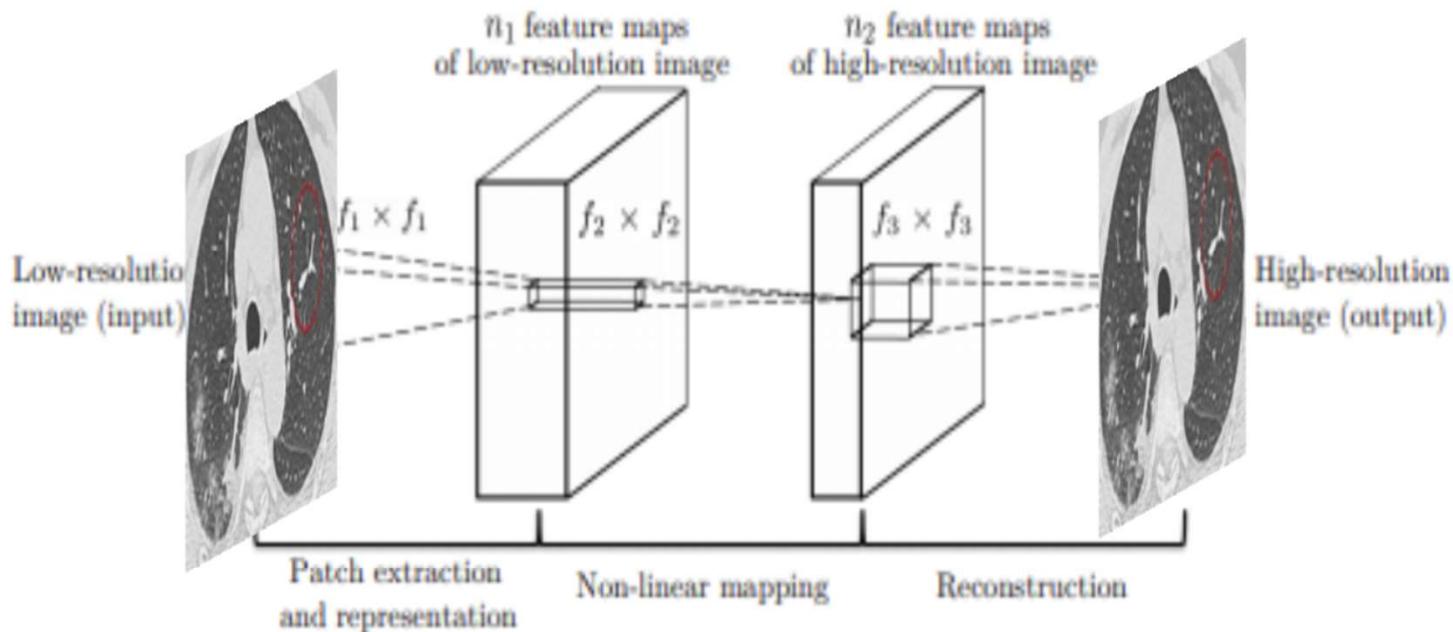
01. Deep Learning

CNN(Convolution Neural Network)

CNN이란 영상, 음성에 주로 사용되는 딥러닝의 한 종류이며,
데이터로부터 자동으로 Feature(특징)을 추출하여 패턴을 파악하는 구조이다.

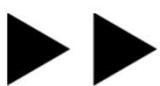
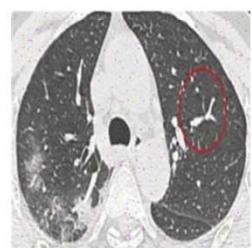


02. SRCNN

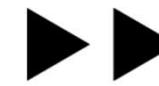


02. SRCNN

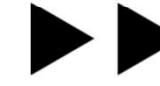
0. Input data 처리



Down sampling



Bicubic interpolation

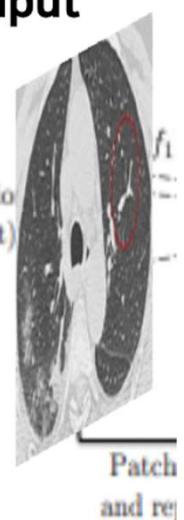


하나의 영상을 더 작은 표본으로
표현하는 과정 위 과정을 통해
화질이 더 저하된다.

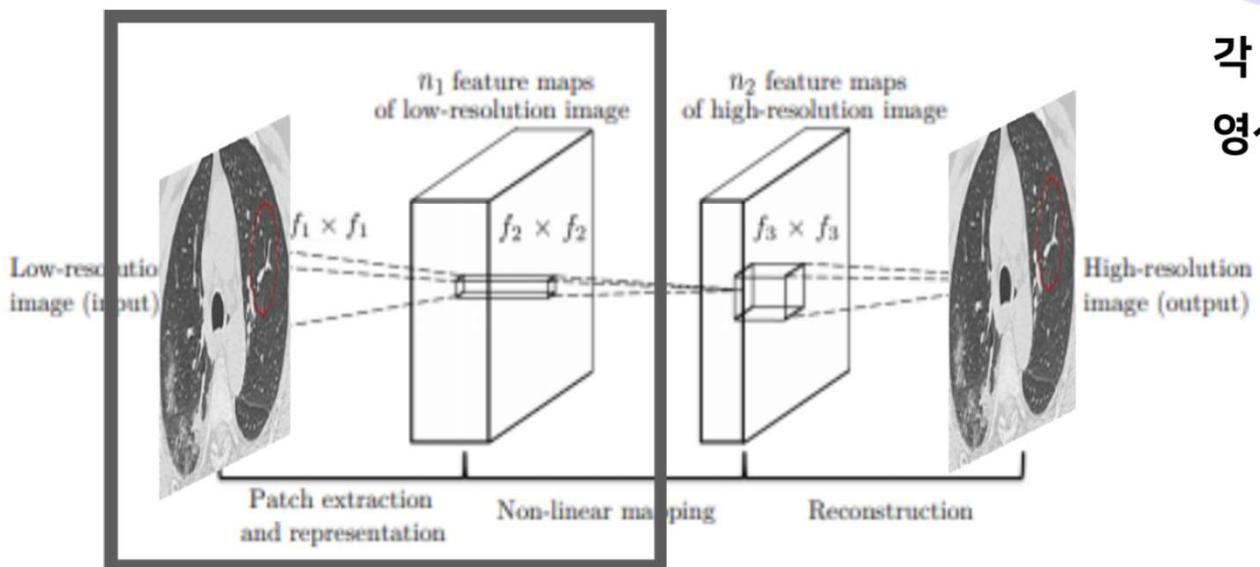
cubic interpolation을 2차원
으로 확장시킨 것으로 네 개의
점을 참조해서 값을 추정해 크기
를 키운다.

input

Low-resolutio
image (input)



02. SRCNN

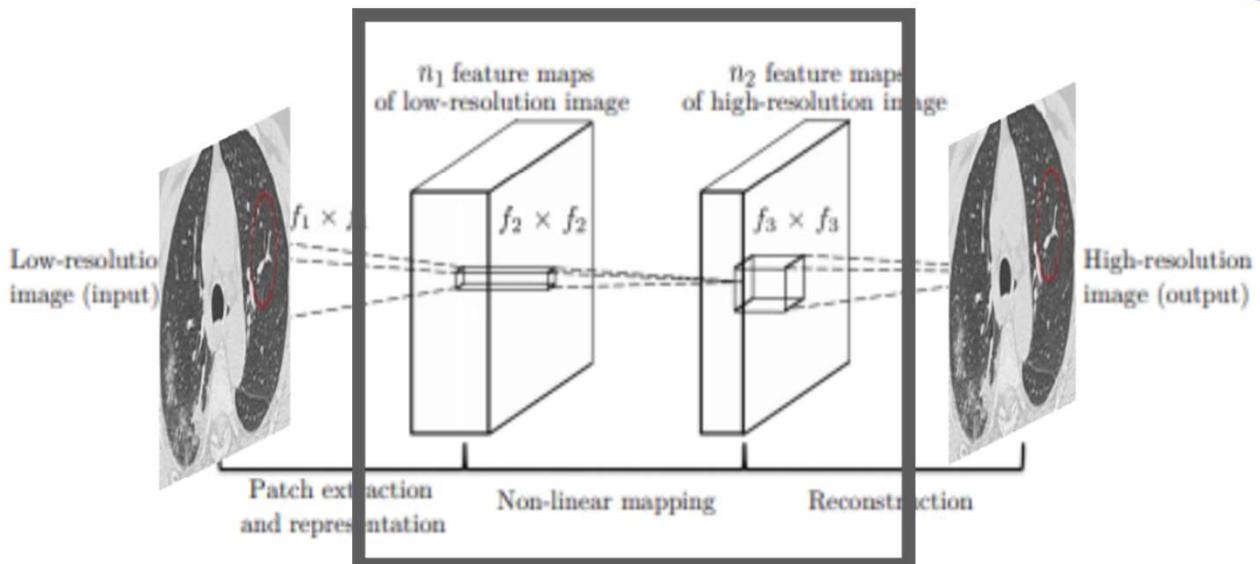


1. Patch extraction and representation

각 픽셀 값을 vector로 표현한다. 위 vector가 영상의 feature map이다.

02. SRCNN

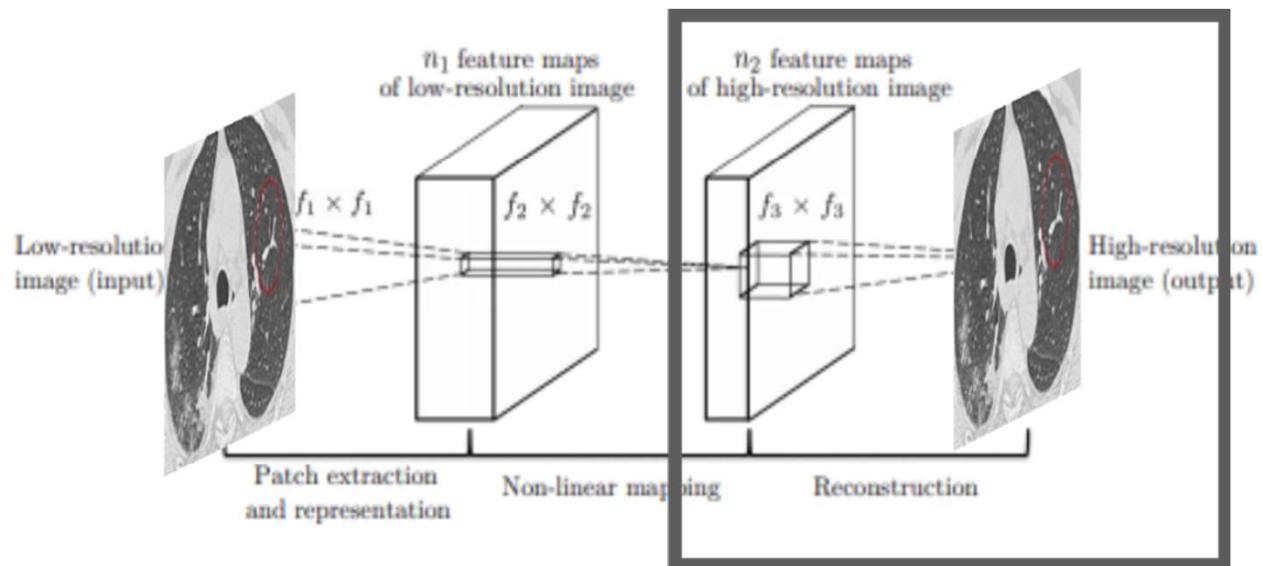
2. Non-Linear mapping



저해상도 이미지에서 고해상도 이미지로 비선형 매핑을 한다.

02. SRCNN

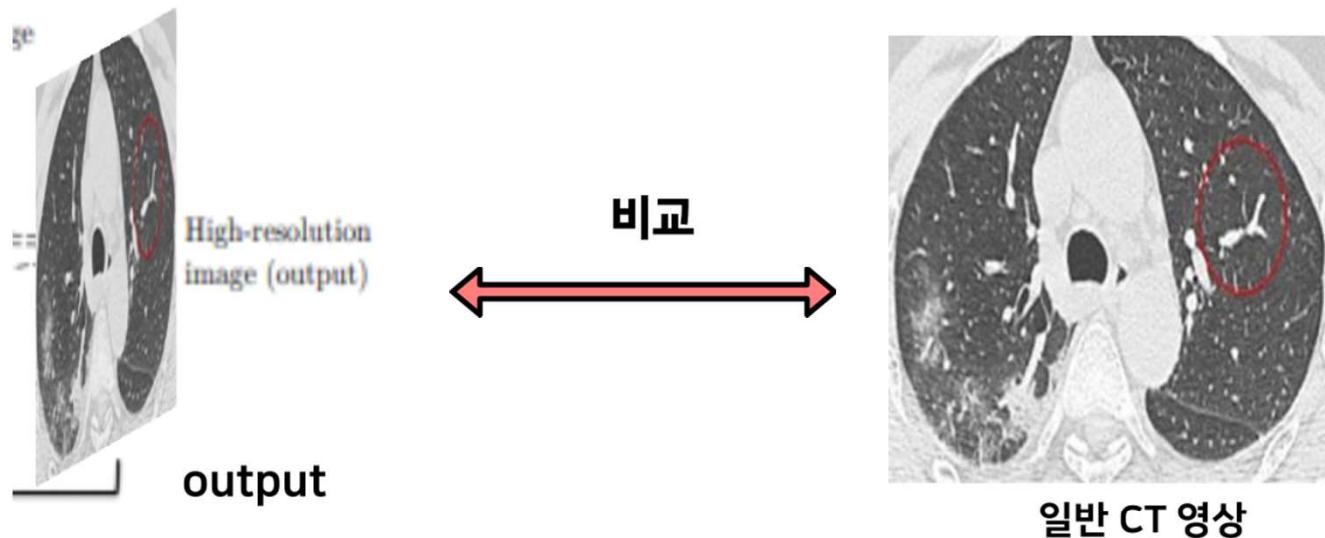
3. Reconstruction



convolution layer로 만들어진 patch
를 모아 고해상도 이미지를 만든다.

02. SRCNN

4. output

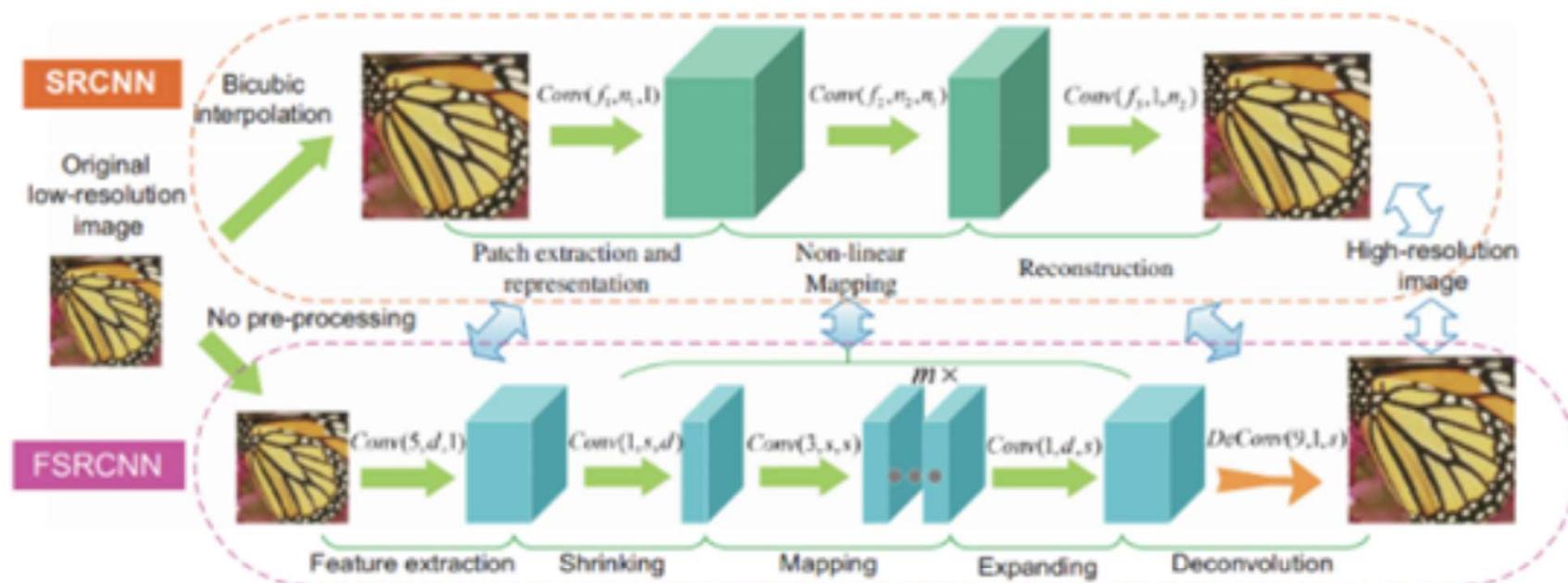


02. SRCNN

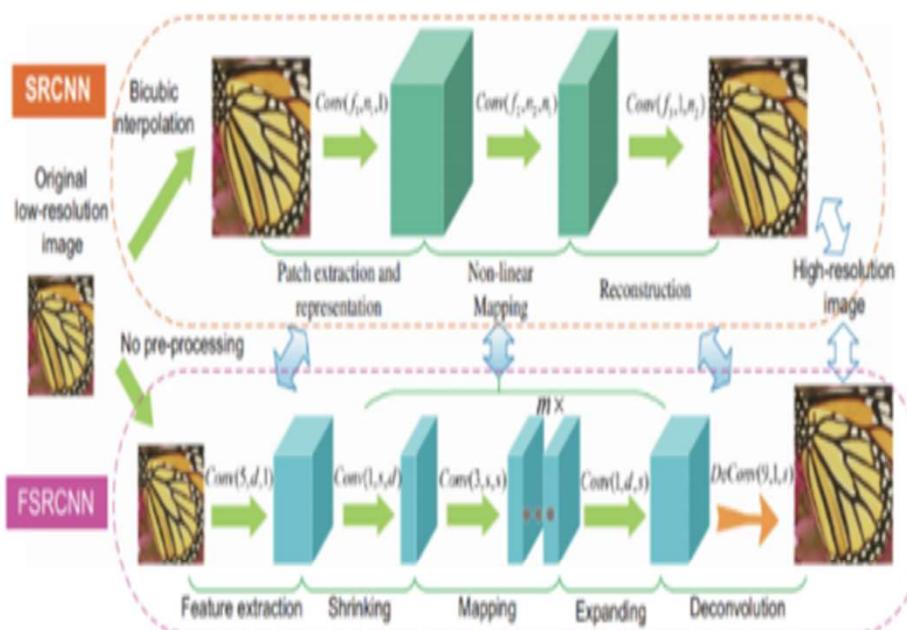
```
import torch.nn as nn
import torch.nn.functional as F
class SRCNN(nn.Module):
    def __init__(self):
        super(SRCNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 64, kernel_size=9, padding=2, padding_mode='replicate')
        self.conv2 = nn.Conv2d(64, 32, kernel_size=1, padding=2, padding_mode='replicate')
        self.conv3 = nn.Conv2d(32, 1, kernel_size=5, padding=2, padding_mode='replicate')
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = self.conv3(x)
    return x
```

loss	BCE Loss
learning rate	0.0001 or 0.00001
optimizer	Adam
batch size	64

03. FSRCNN



03. FSRCNN



1) SRCNN에서 interpolation으로 인한 다양한 연산량을 방지하기 위해 별도의 전처리 없이 input에 넣어준다.

2) 5 단계로 이루어져 있지만 SRCNN과 처리 하는 layer의 일은 동일하다.

3) SRCNN과 비교하기 위해 만들어진 모형이기 때문에 마지막 output의 size가 같아야 한다.
따라서 Deconvolution layer를 추가하여 size를 키운다.

03. FSRCNN

```
class FSRCNN(nn.Module):
    def __init__(self, scale_factor, num_channels=1, d=56, s=12, m=4):
        super(FSRCNN, self).__init__()
        self.first_part = nn.Sequential(
            nn.Conv2d(num_channels, d, kernel_size=5, padding=5//2),
            nn.PReLU(d)
        )
        self.mid_part = [nn.Conv2d(d, s, kernel_size=1), nn.PReLU(s)]
        for _ in range(m):
            self.mid_part.extend([nn.Conv2d(s, s, kernel_size=3, padding=3//2), nn.PReLU(s)])
        self.mid_part.extend([nn.Conv2d(s, d, kernel_size=1), nn.PReLU(d)])
        self.mid_part = nn.Sequential(*self.mid_part)
        self.last_part = nn.ConvTranspose2d(d, num_channels, kernel_size=9, stride=scale_factor, padding=9//2,
                                         output_padding=scale_factor-1)

        self._initialize_weights()

    def _initialize_weights(self):
        for m in self.first_part:
            if isinstance(m, nn.Conv2d):
                nn.init.normal_(m.weight.data, mean=0.0, std=math.sqrt(2/(m.out_channels*m.weight.data[0][0].numel())))
                nn.init.zeros_(m.bias.data)
        for m in self.mid_part:
            if isinstance(m, nn.Conv2d):
                nn.init.normal_(m.weight.data, mean=0.0, std=math.sqrt(2/(m.out_channels*m.weight.data[0][0].numel())))
                nn.init.zeros_(m.bias.data)
        nn.init.normal_(self.last_part.weight.data, mean=0.0, std=0.001)
        nn.init.zeros_(self.last_part.bias.data)

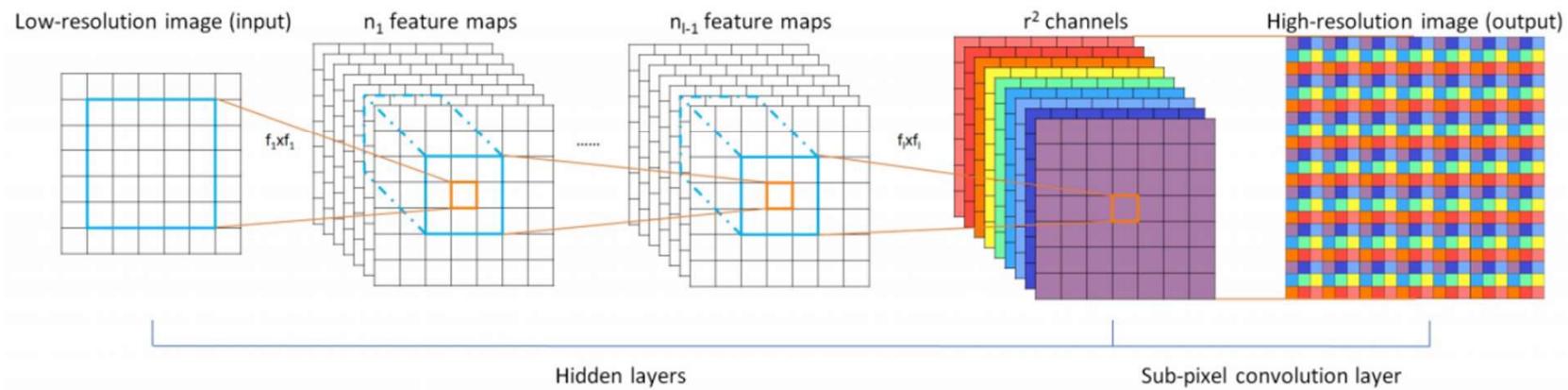
    def forward(self, x):
        x = self.first_part(x)
        x = self.mid_part(x)
        x = self.last_part(x)
        return x
```

loss	BCE Loss
learning rate	0.01
optimizer	Adam
batch size	16

04. ESPCN

ESPCN은 FSRCNN과 모형 구조 대부분이 흡사하다.

다른 점은, Deconvolution layer이 아닌 Sub-pixel convolution layer를 사용해 고화질 영상을 얻는다.



04. ESPCN

```
class ESPCN(nn.Module):
    def __init__(self, scale_factor, num_channels=1):
        super(ESPCN, self).__init__()
        self.first_part = nn.Sequential(
            nn.Conv2d(num_channels, 64, kernel_size=5, padding=5//2),
            nn.Tanh(),
            nn.Conv2d(64, 32, kernel_size=3, padding=3//2),
            nn.Tanh(),
        )
        self.last_part = nn.Sequential(
            nn.Conv2d(32, num_channels * (scale_factor ** 2), kernel_size=3, padding=3 // 2),
            nn.PixelShuffle(scale_factor)
        )

        self._initialize_weights()

    def _initialize_weights(self):
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                if m.in_channels == 32:
                    nn.init.normal_(m.weight.data, mean=0.0, std=0.001)
                    nn.init.zeros_(m.bias.data)
                else:
                    nn.init.normal_(m.weight.data, mean=0.0, std=math.sqrt(2/(m.out_channels*m.weight.data[0][0].numel())))
                    nn.init.zeros_(m.bias.data)

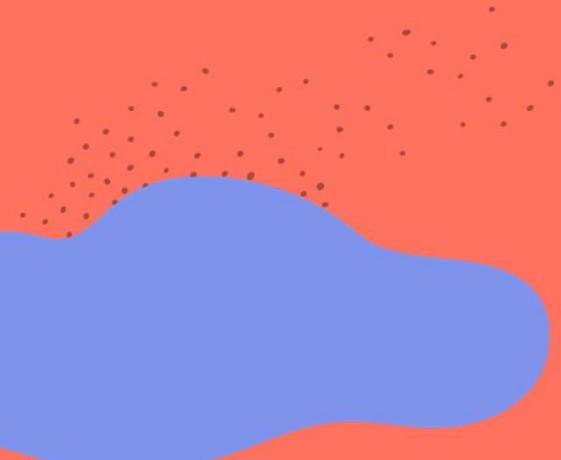
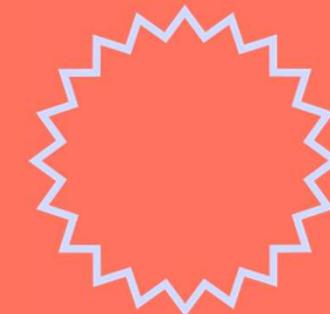
    def forward(self, x):
        x = self.first_part(x)
        x = self.last_part(x)
        return x
```

loss	MSE
learning rate	0.1
optimizer	Adam
batch size	16

04



01. 모델 결과
02. 결론 및 아쉬운 점
03. 출처



01. 모델 결과

생성한 고화질 CT 영상이 잘 만들어졌는지 원본 CT 영상과 비교 하는 판단 척도는 PSNR이다.
PSNR(Peak Signal to Noise Ratio)은 최대 신호 대 잡음비로 구하는 방법은 다음과 같다.

```
def psnr(img1, img2):
    mse = np.mean((img1 - img2)**2)

    if mse == 0:
        return 100
    else:
        return 20 * math.log10(255.0 / math.sqrt(mse))
```

$$PSNR = 20 \times \log_{10}(MAX_y) - 10 \times \log_{10}(MSE_y)$$

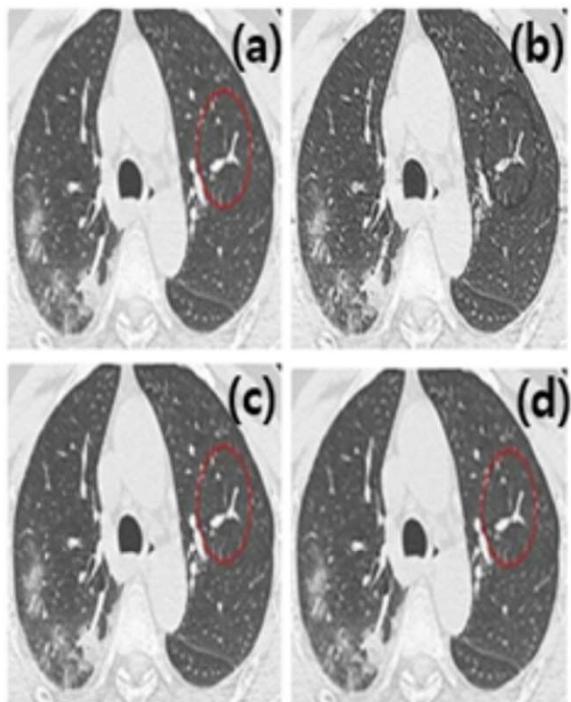
MAX : 모형을 통해 획득한 최대 픽셀값

MSE : 영상의 각 픽셀에 대한 예측값과 실제값의 차이의 제곱

PSNR 값이 높을수록 재구성된 이미지의 품질이 좋다는 의미이다.

01. 모델 결과

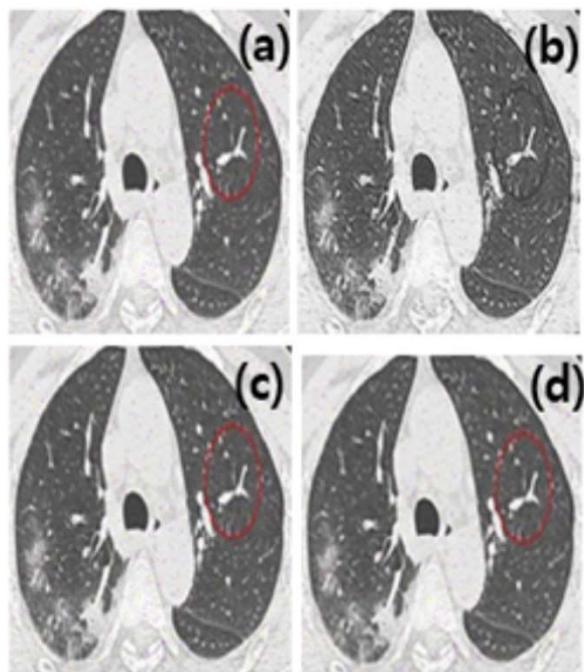
1) Gaussian Noise



모형	PSNR
(b) SRCNN	30.79
(c) FSRCNN	36.02
(d) ESPCN	30.11

01. 모델 결과

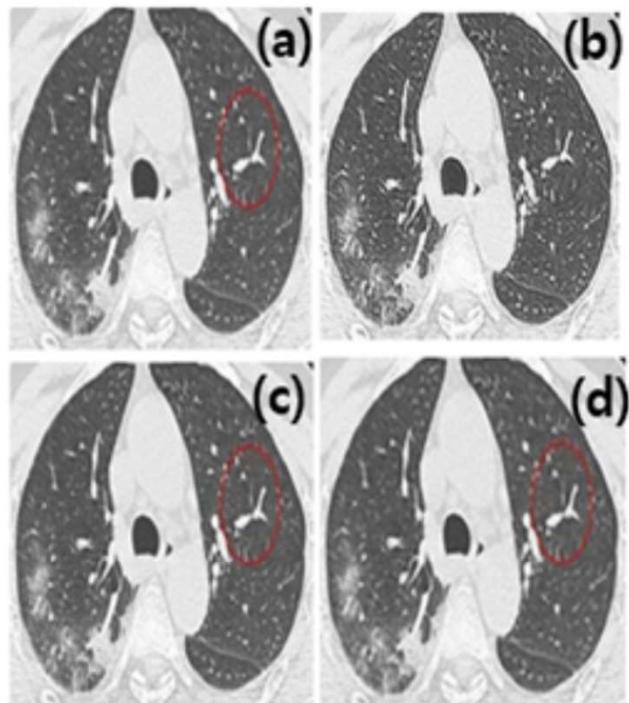
2) S&P Noise



모형	PSNR
(b) SRCNN	30.13
(c) FSRCNN	31.10
(d) ESPCN	27.71

01. 모델 결과

3) Poisson Noise



모형	PSNR
(b) SRCNN	30.11
(c) FSRCNN	35.74
(d) ESPCN	29.77

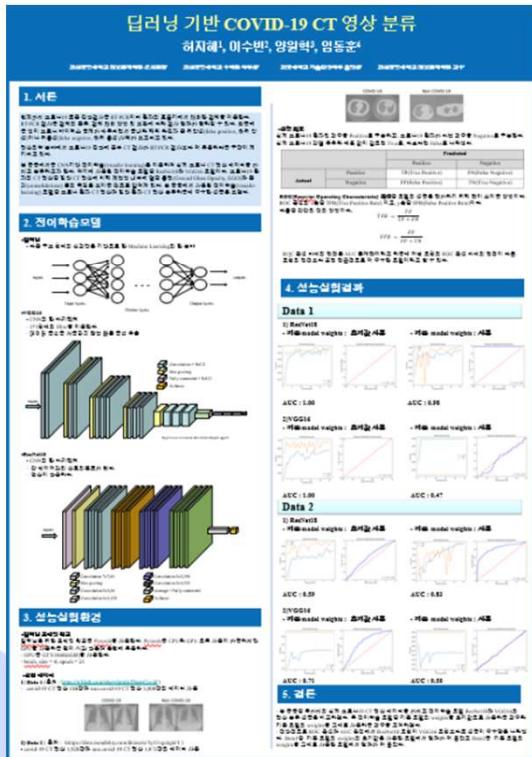
02. 결론 및 아쉬운 점

결론

- 저화질 영상을 고화질 영상으로 만든다는 뜻은 Noise를 영상 복원으로 제거가 가능함을 뜻한다.
- 세 가지 Noise에 대하여 측정해보니 FSRCNN이 모든 Noise에서 가장 좋은 PSNR 값을 나타냄을 보였다.
- 가지고 있는 데이터에 맞게 여러 파라미터를 조절하면 더욱 높은 결과값을 얻을 수 있을 것이다.

02. 결론 및 아쉬운 점

연구 성과



포스터 발표(5월 28일)
한국통계학회 포스터
발표에 참여하여 발표를
진행 하였다.

아쉬운 점

- 처음 연구 계획이랑 방향성이 달라져 관련 논문을 학회에 제출을 못해 아쉽다.
- 위 논문에 나온 모형 말고도 VDSR, U-Net 등의 모형을 공부했는데 구현에 실패하여 아쉽다.
- 전문가 초빙을 논문 제출 막판에 하여 수정을 많이 못 한게 아쉬웠다.

03. 출처



<https://github.com/UCSD-AI4H/COVID-CT>

<https://github.com/yjn870/SRCNN-pytorch>

<https://github.com/yjn870/FSRCNN-pytorch>

<https://github.com/yjn870/ESPCN-pytorch>

Dong, Chao, et al. "Image super-resolution using deep convolutional networks." *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2015): 295-307.

Dong, Chao, Chen Change Loy, and Xiaoou Tang. "Accelerating the super-resolution convolutional neural network." *European conference on computer vision*. Springer, Cham, 2016.

e, et al. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

"Computed Tomography Using Super-resolution Convolutional Neural Network", for *Journal of the Korean Society of Radiology* (2020) Volume 14 Issue 3 211-220

"Ultra-low-dose chest CT imaging of COVID-19 patients using a deep residual neural network", proceeding for *European Radiology* (2021) 31-1420-1431

"Classification of COVID-19 pneumonia from chest CT images based on reconstructed super-resolution images and VGG neural network", for *Health Inf Sci Syst*, (2021) 10 100
7/s13755-021-00140-0

CT 영상에서 iNCEPTION 모형을 이용한 잡음제거". for *Journal of the Korean Data Information Science Society*, (2020), 31(2), 487-501

THANK YOU

발표를 들어주셔서 감사합니다 :)

