

Image Restoration Report

학번 : 2019202032

이름 : 이상현

개발환경

Keras & colab (+tensorflow(tensorflow_datasets))

과제 개요

주어진 3000개의 train image data와 100개의 test image data에 대해 노이즈를 생성하고, 여러 방법을 통해 노이즈를 제거한 뒤, 각 방법에서의 psnr을 비교한다. 이때 노이즈는 가우시안 노이즈와 salt&pepper 노이즈를 생성하며 노이즈를 제거할 때는 alpha-trimmed mean filtering 과 autoencoder를 사용한다. Alpha-trimmed mean filtering을 통해 노이즈를 제거할 때는 filter size를 3x3, 5x5로 2개를 다르게 하여 진행하고, autoencoder를 통해서 노이즈를 제거할 때는 32channels 과 64channels 두 개를 사용하여 제거한다.

과제 수행 방법

1) 노이즈 생성

- gaussian noise

가우시안 노이즈를 생성하기 위해서 표준편차가 10, 평균이 0인 가우시안 분포의 데이터를 랜덤으로 256x256크기의 배열에 저장한다. 저장한 배열과 주어진 이미지 데이터를 각 픽셀단위로 더한다. 이때 더한 값이 0보다 작다면 해당 픽셀의 값을 0으로, 255보다 크다면 해당 픽셀의 값을 255로 고정하도록 한다.

- salt & pepper noise(s&p)

s&p 노이즈를 생성하기 위해서 가장 먼저 해야할 것은 노이즈를 추가할 픽셀의 위치를 이미지에서 고르는 것이었다. 따라서 중복순열 알고리즘을 활용하여 x와 y의 값으로 이루어진 (x, y)형태의 값을 모든 픽셀의 수 256x256개의 20%에 해당하는 13107개를 추출하였다. 이후 13107개의 랜덤한 위치에 0 또는 255의 값을 랜덤으로 추가하여 노이즈를 생성하였다.

2) alpha-trimmed mean filtering

- filter size : 3x3, alpha = 0.2

먼저 filter size가 3x3이므로 이미지를 순회하며 filter에 해당하는 값을 배열에 담아준다. 이때 측면에 해당하는 부분의 경우 값이 부족하기 때문에 zero padding을 해주었고, 따라서 배열에 담을 때는 해당 위치의 값을 0으로 대체하였다. 이후, 9개의 값이 모두 담겼다면, 해당 배열을 정렬하고, alpha = 0.2를 제거한 7개의 데이터의 평균값을 계산하여 대응하는 픽셀에 저장한다.

- filter size : 5x5, alpha = 0.4

먼저 filter size가 5x5이므로 이미지를 순회하며 filter에 해당하는 값을 배열에 담아준다. 이때 측면에 해당하는 부분의 경우 값이 부족하기 때문에 zero padding을 해주었고, 따라서 배열에 담을 때는 해당 위치의 값을 0으로 대체하였다. 이후, 25개의 값이 모두 담겼다면, 해당 배열을 정렬하고, alpha = 0.4를 제거한 5개의 데이터의 평균값을 계산하여 대응하는 픽셀에 저장한다.

3) Autoencoder

(1) Sequential Model

Convolution

- relu함수를 이용하여 데이터의 convolution을 수행한다.
- maxpooling을 통해 데이터의 크기를 줄여 메모리의 차지하는 양을 줄인다.
- relu함수를 이용한 convolution을 다시 수행한다.
- maxpooling을 통해 64x64 크기의 데이터로 만든다.

Convolution transpose

- stride를 2로 하고 relu함수를 2번 사용하여 원래의 이미지로 생성한다.
- sigmoid 함수를 사용하여 출력 값을 0에서 1 사이의 값으로 제한한다.

(2) compile을 통해 모델 학습 과정을 설정한다.

(3) fit을 통해 주어진 epoch수 만큼 모델을 학습시킨다.

4) psnr

Numpy의 mean함수를 사용해 원본 이미지와 원복한 이미지 픽셀 값의 차의 제곱의 평균을 계산한다. 이후 아래의 psnr 공식에 맞춰 psnr 값을 계산한다.

- normalized data

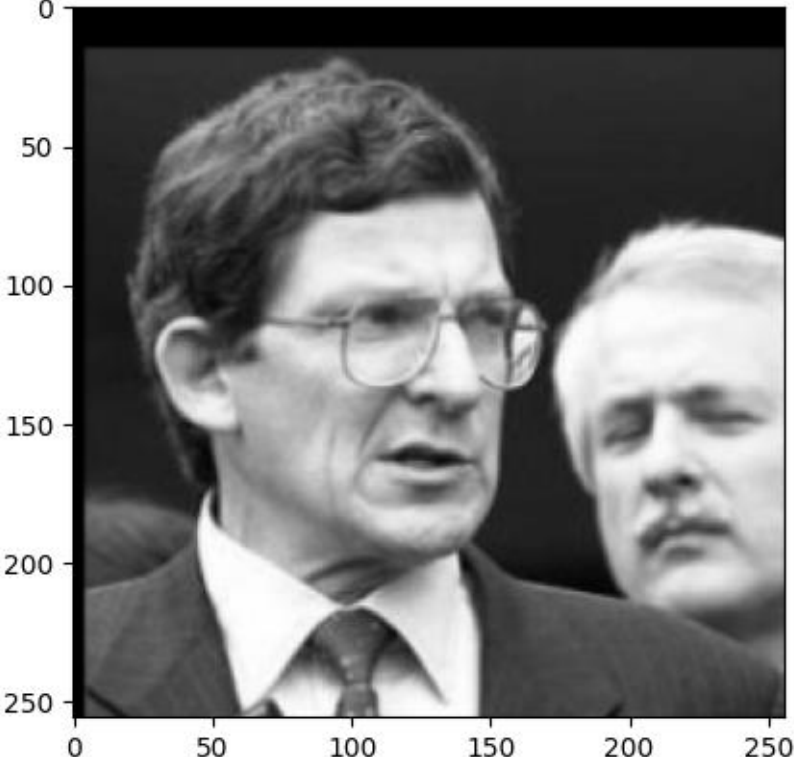
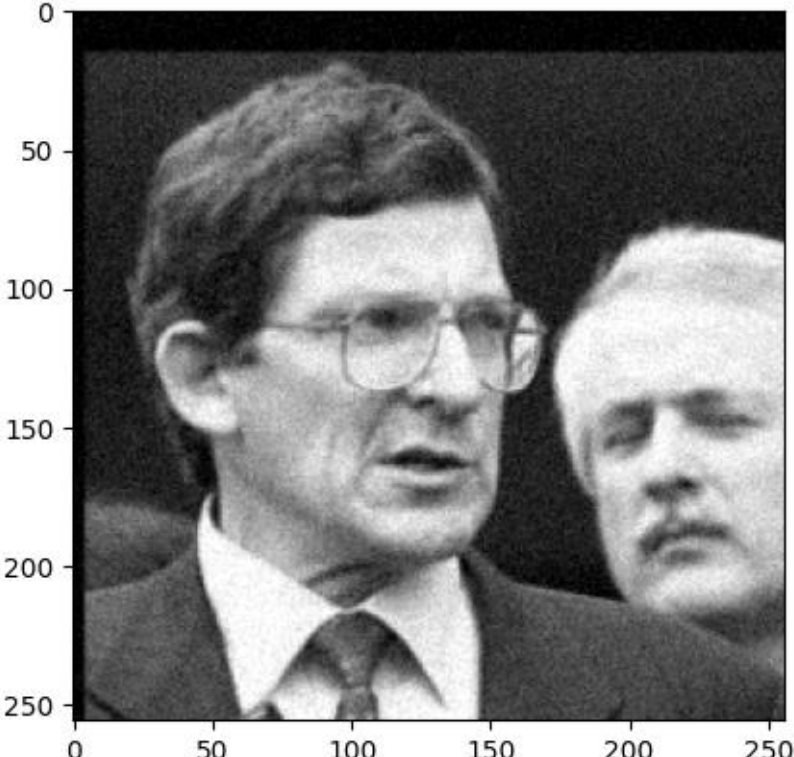
$$\text{psnr} = 10 * \text{np.log10}(1 / (\text{mse}))$$

- not_normalized data

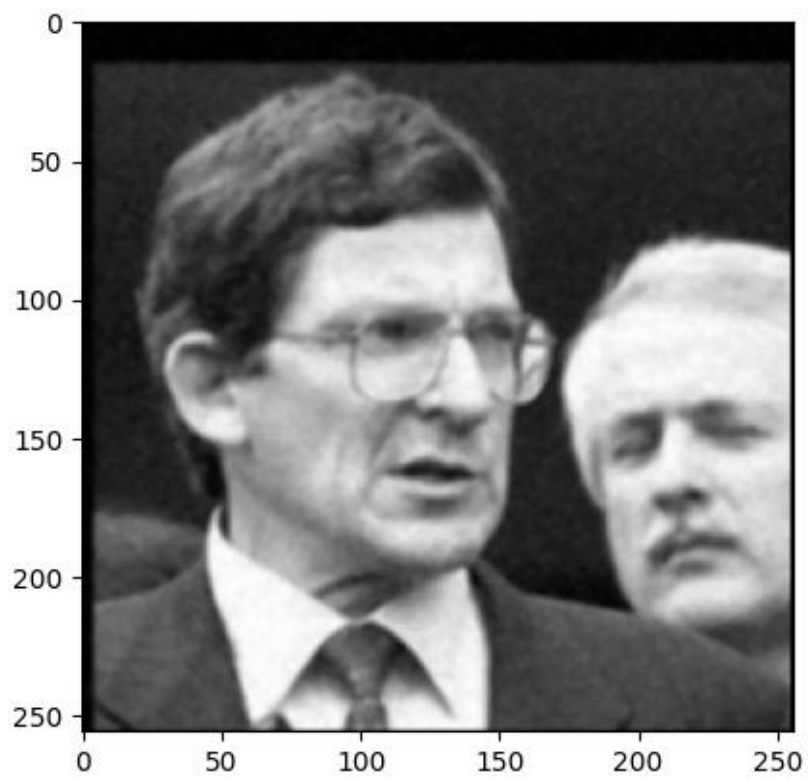
$$\text{psnr} = 10 * \text{np.log10}(255 * 255 / (\text{mse}))$$

결과 분석

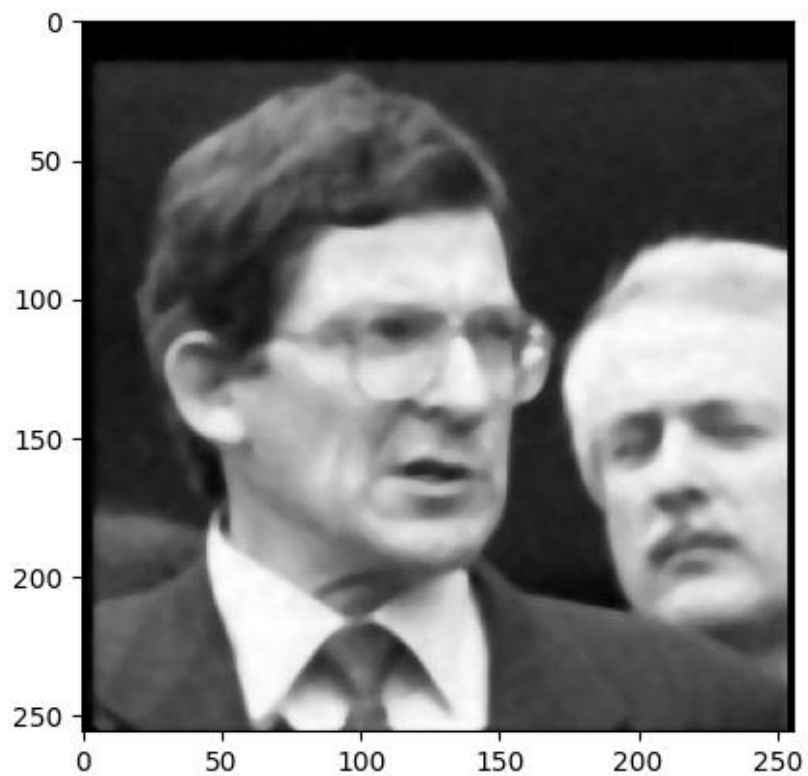
<test data 1>

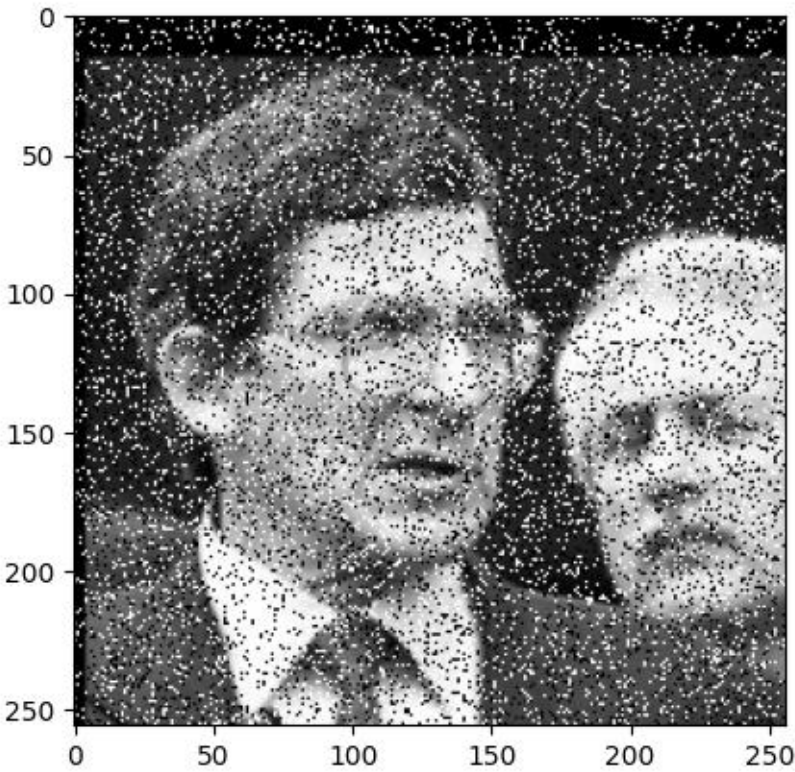
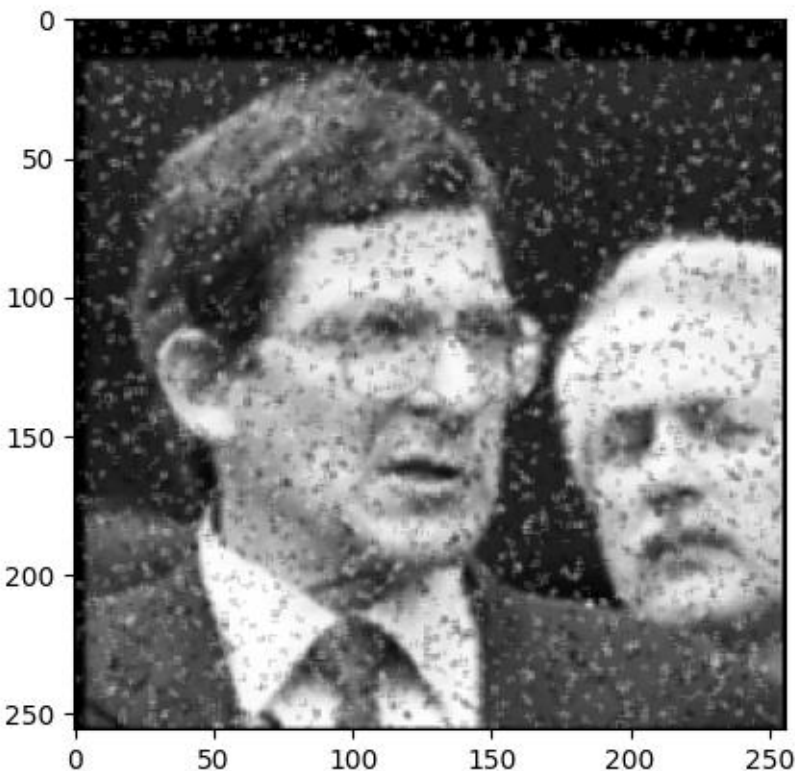
Original data	
가우시안 노이즈	



3x3 alpha
trimmed
Denoised 가
우시안



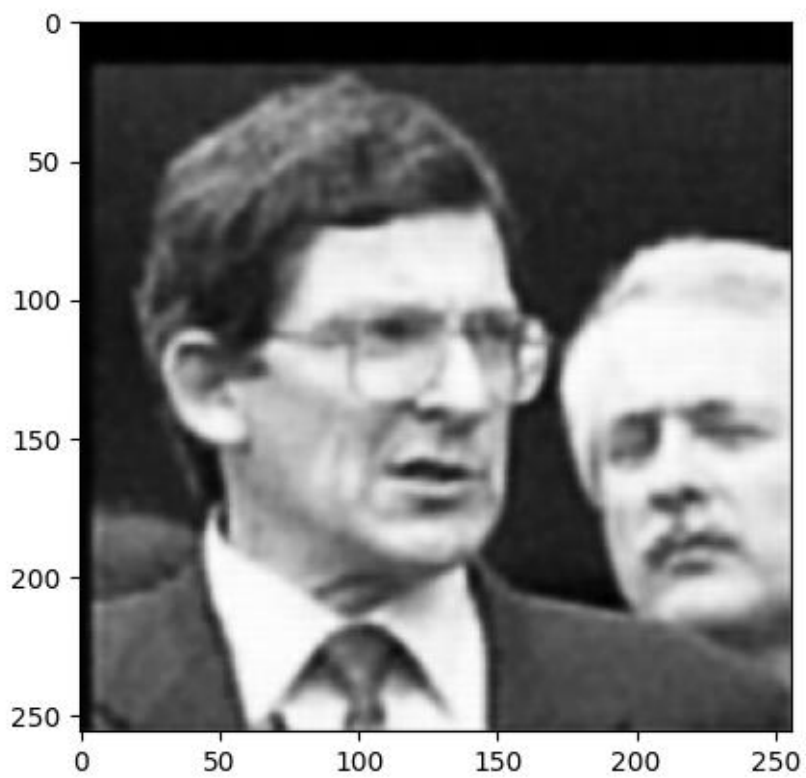
5x5 alpha
trimmed
Denoised 가
우시안



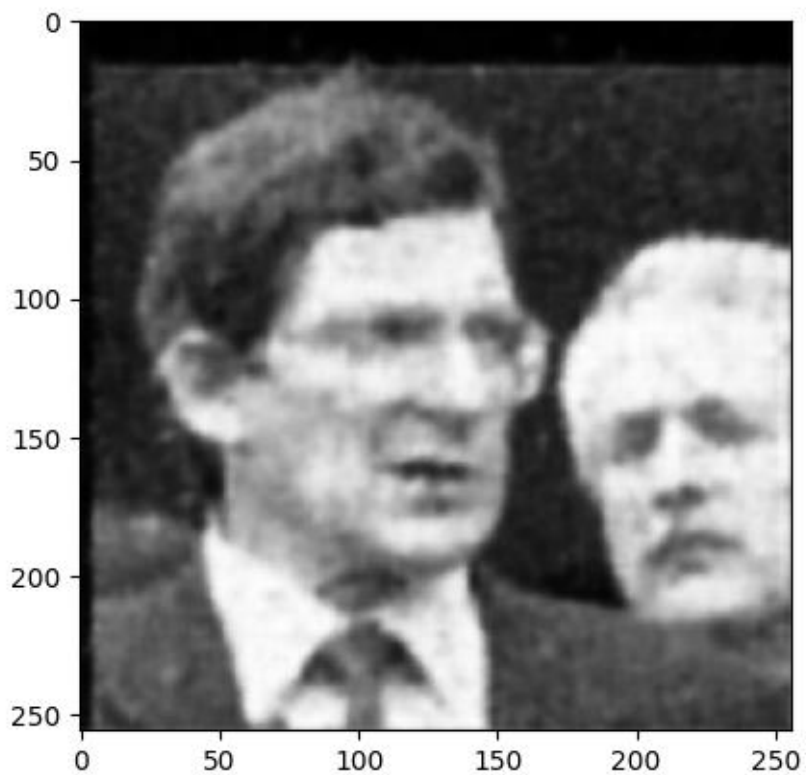
<p>Salt and pepper</p>	
<p>3x3 alpha trimmed Denoised S&P</p>	

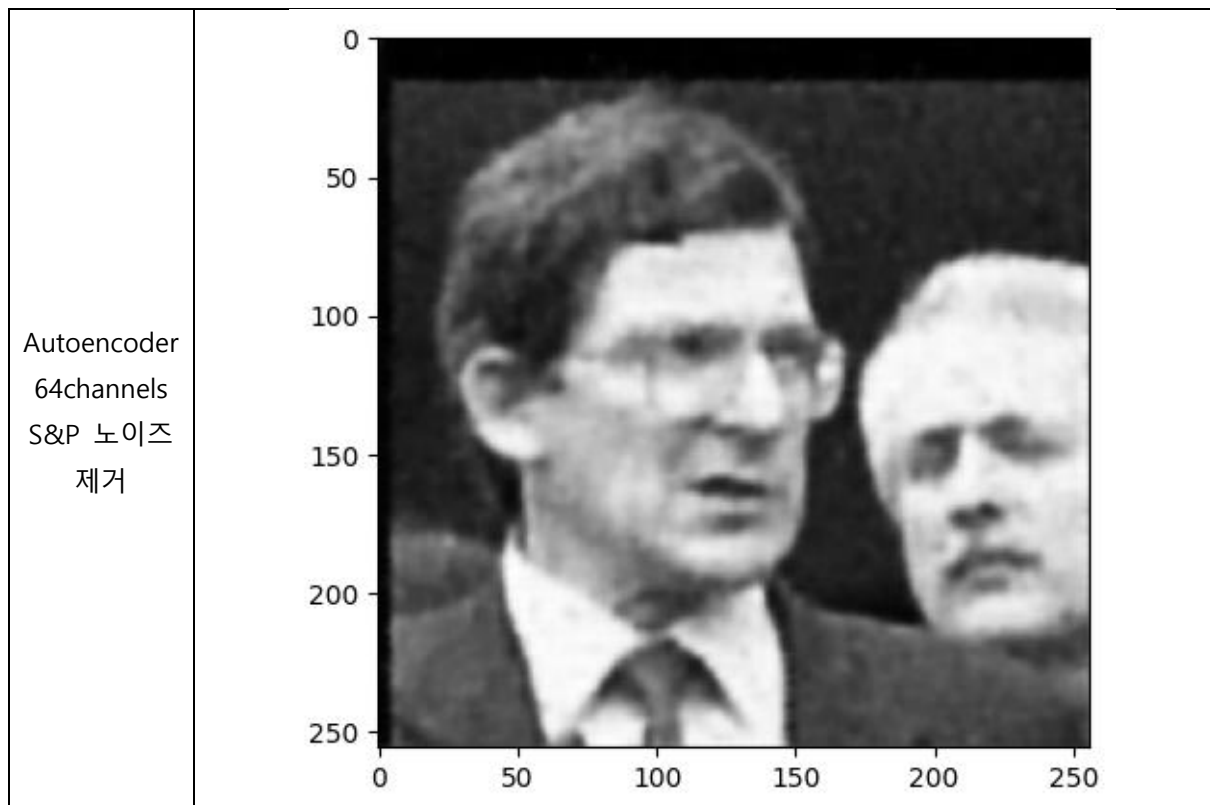
<p>5x5 alpha trimmed Denoised S&P</p>	 <p>A 256x256 grayscale image of a man with glasses and a suit, with another person partially visible in the background. The image is processed with 5x5 alpha trimmed denoising and S&P filtering. The image shows a man with dark hair and glasses, wearing a dark suit, white shirt, and dark tie. He is looking slightly to the right. Another person with light hair is visible in the background to the right. The image has a slightly grainy texture, characteristic of denoising.</p>
<p>Autoencoder 32channels 가우시안 노 이즈 제거</p>	 <p>A 256x256 grayscale image of the same man and background as the top image, but processed with an autoencoder with 32 channels and Gaussian noise removal. The image appears slightly more refined than the top one, with less visible noise and a smoother texture. The man's features are clearly visible, and the background is also well-defined.</p>

Autoencoder
64channels
가우시안 노
이즈 제거



Autoencoder
32channels
S&P 노이즈
제거





Test1 데이터에 대해서 위와 같은 결과를 얻을 수 있었다. 육안으로 확인했을 때 가우시안 노이즈의 경우 5x5와 3x3 filter에 큰 차이가 없어 보였지만, salt and pepper 노이즈의 경우 5x5와 3x3 filter의 차이가 크게 보이는 것을 확인할 수 있었다. 이는 salt and pepper 노이즈의 경우 가우시안 노이즈와 비교했을 때 원본의 픽셀 값을 유지하고 있는 데이터가 많고 이에 따라 중간 값을 사용하여 이미지를 원복할 경우 더 많은 원본 픽셀값을 사용할 수 있어 필터의 크기가 클수록 가우시안에서 필터의 크기가 커지는 것 대비 더 많은 원복 효과를 누릴 수 있는 것으로 생각되었다.

<PSNR 비교>

```

psnr_gaussian_denoise_data_5x5 : 34.7586225436027
psnr_gaussian_denoise_data_3x3 : 35.099968550756245
psnr_snp_denoise_data_5x5 : 35.76373084312472
psnr_snp_denoise_data_3x3 : 31.625567034084405
psnr_auto_predicted_gaussian_noised_test_data_32 : 31.558552133614512
psnr_auto_predicted_gaussian_noised_test_data_64 : 32.49399620615166
psnr_auto_predicted_snp_noised_test_data_32 : 27.065861640387
psnr_auto_predicted_snp_noised_test_data_64 : 29.598045480981252

```

Parameter setting	Alpha-trimmed mean filtering		Autoencoder	
	Filter 3x3, a = 0.2	Filter 5x5, A = 0.4	32channels	64channels
Gaussian noise	35.099db	34.758	31.558db	32.493db
Salt & pepper noise	31.625db	35.763db	27.065db	29.598db

가우시안 노이즈를 제거할 때는 필터의 크기가 크거나 channel이 더 크더라도, 즉 연산량이 더 많을 경우에도 오히려 psnr의 값이 더 작아지는 경우가 있음을 확인할 수 있고, salt & pepper 노이즈를 제거할 때는 필터의 크기가 크거나 채널의 크기가 클 경우 psnr이 더 커지는 것을 확인할 수 있다. 이는 개인적으로 salt&pepper 노이즈가 가우시안 노이즈에 비해 원본 이미지를 더 많이 훼손하기 때문이라고 생각하였다. Salt&pepper 노이즈의 경우 특정 픽셀의 값을 0이나 255로 바꿔버리기 때문에 주변 값을 더 많이 이용할수록, 연산량이 많을수록 원본 이미지에 더 가깝게 원복할 수 있다고 생각하였다. 반면 가우시안 노이즈의 경우 원본 이미지의 손상자체가 크지 않기 때문에 필터 사이즈나 채널의 수에 큰 영향을 받지 않는다고 생각되었다. 만약 가우시안 노이즈를 생성하는 과정에서 표준편차를 더 크게 설정하여 노이즈를 더 크게 만든다면 salt&pepper 노이즈를 제거할 때와 마찬가지로 더 많은 연산을 할수록 더 높은 psnr을 가지게 될 것이라고 생각되었다. 이를 통해 원본이미지에서 훼손이 많이 되었다면 더 많은 연산을 하는 것이 중요하고, 반대로 훼손이 크지 않다면 연산량과 회복정도의 trade-off를 고려하는 것이 중요하다고 생각되었다.

고찰

가우시안 노이즈를 생성하여 원본 이미지에 더할 때 특정 픽셀에서 색이 아예 검은색이나 흰색으로 나오는 경우가 발생하였다. 특히 머리카락처럼 검은 부분이나 흰 부분에서 그러한 현상이 자주 발생하였는데, 이는 원본 이미지의 픽셀값이 0이나 255에 가깝고, 거기에 가우시안 노이즈의 값을 더하니 underflow나 overflow가 발생하여 생기는 현상이었다. 이러한 문제를 해결하기 위하여 가우시안 노이즈의 값과 원본 픽셀에서의 값을 더해서 바로 저장하는 것이 아니라 따로 int형 변수에 먼저 더하고, 해당 값이 0이하 혹은 255이상이면 따로 처리하는 방식을 통해 해결하였다. 이를 통해 이미지를 다룰 때는 underflow나 overflow를 조심해야한다는 것을 알게되었고, 동시에 코드의 순서를 조금만 바꾸거나 예외처리를 조금만 해도 이를 예방할 수 있다는 것을 알게되었다.

이번 과제를 통해 cnn 인공지능 학습 모델을 처음 설계하고 사용해보았다. 해당 과제를 통해 각 모델에서 layer를 몇 개 사용할 것인지, 각 layer에 어떤 함수를 사용할지에 따라 결과가 바뀐다는 것을 알게되었고, 또한 무엇보다 학습하기 위해 주어지는 데이터가 중요하다는 생각을 하게되었다. 뿐만 아니라 overfitting이 되지 않는 한에서 적절한 횟수의 학습 횟수를 설정하는 것과 batch

size를 얼마나 할 것인지 또한 중요하다는 생각을 하게 되었다. 특히 학습 횟수에 따라서 loss율이 감소하는 것을 확인할 수 있었기 때문에 적절한 횟수의 학습량을 선택할 필요가 있다는 것을 알게되었다. 따라서 이후에도 다양한 모델을 설계하고, 이를 통해 여러 작업을 수행하면서 어떻게 cnn모델을 잘 설계할 수 있는지 등에 대해 알아봐야겠다는 생각을 가지게 되었다.