[과제2]

Image Restoration

2024-04-01 (월)

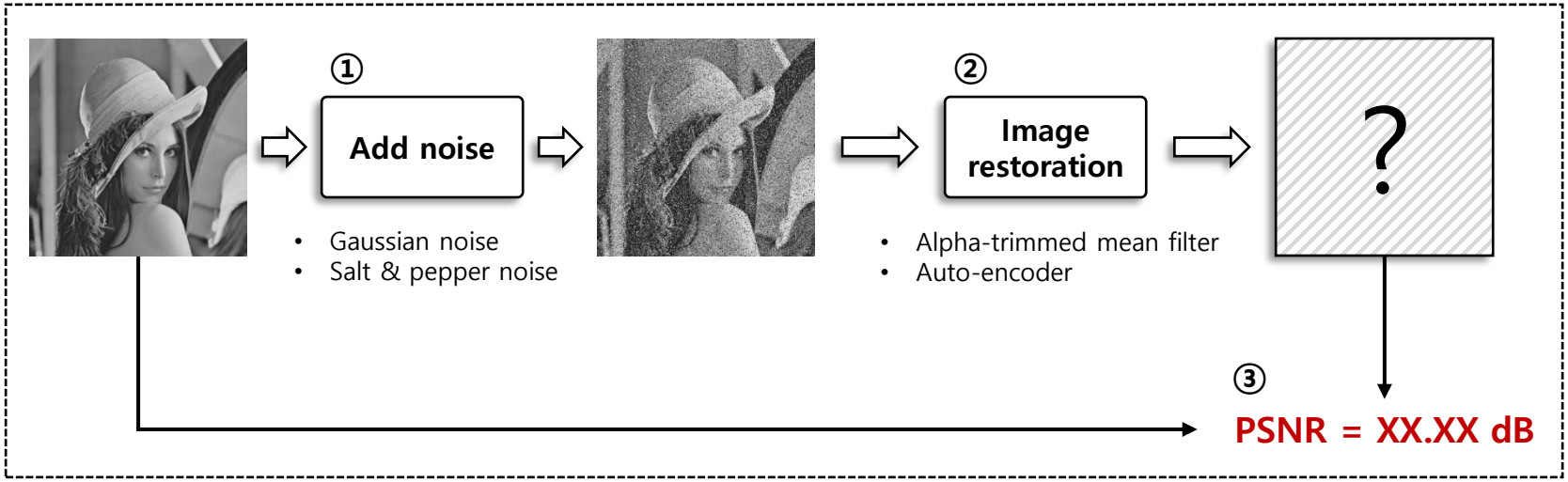
@ 2024-1 컴퓨터비젼

Contents

- 과제 개요
- Noise
  - Additive Gaussian noise
  - Salt & pepper noise
- Image restoration
  - Alpha-trimmed mean filtering
  - Denoising Auto-Encoder
- 과제 세부 사항
- Python 개발 환경
  - Anaconda + Visual Studio Code
  - Google Colab
  - Tensorflow Keras

과제 개요

1. 잡음을 생성하여 입력 이미지에 추가
   - Gaussian noise
   - Salt & pepper noise
2. 잡음이 추가된 이미지에 대해 잡음 제거 알고리즘 수행
   - Alpha-trimmed mean filter
   - Auto-encoder
3. PNSR을 측정하여 잡음 제거 성능 비교



NOISE

Noise

- Noise
  - 영상 획득 과정에서 출력 영상의 픽셀 값이 원본과는 다른 픽셀 값으로 변경되는 것

- 대표적인 Noise 종류
  - Additive Gaussian noise
    - 잡음 값의 분포가 가우시안 분포로 나타나는 잡음
      - ➢ pdf (probability distribution function)

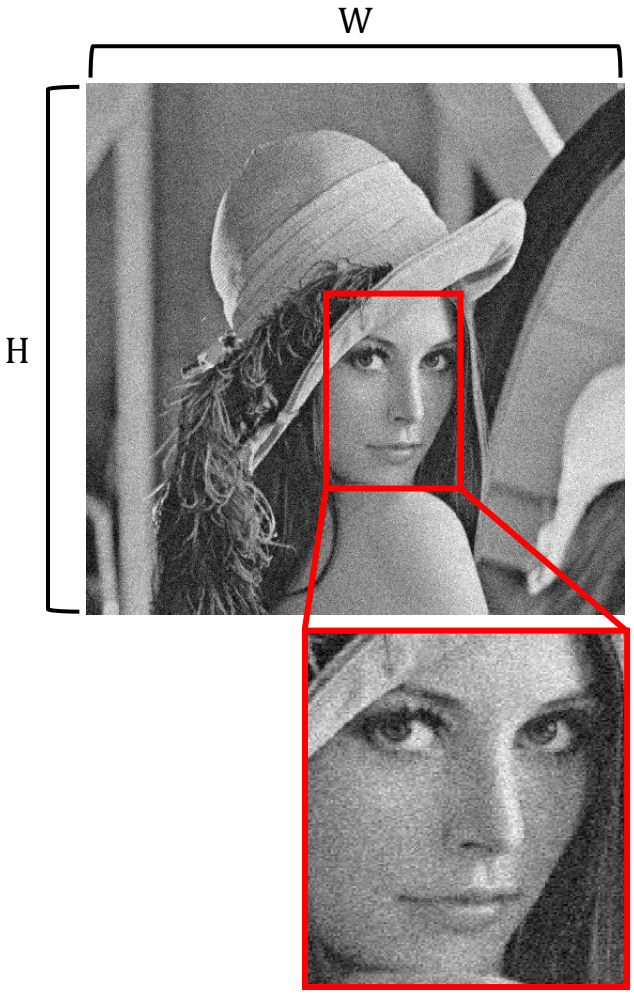
$$p(n) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(n-\mu)^2}{2\sigma^2}}$$

  - Salt & pepper noise
    - 무작위적인 위치에 희고 검은 점이 나타나는 잡음
      - ➢ pdf (probability distribution function)

$$p(n) = \begin{cases} P_a & n = a \\ P_b & n = b \\ 0 & otherwise \end{cases}$$

Additive Gaussian noise

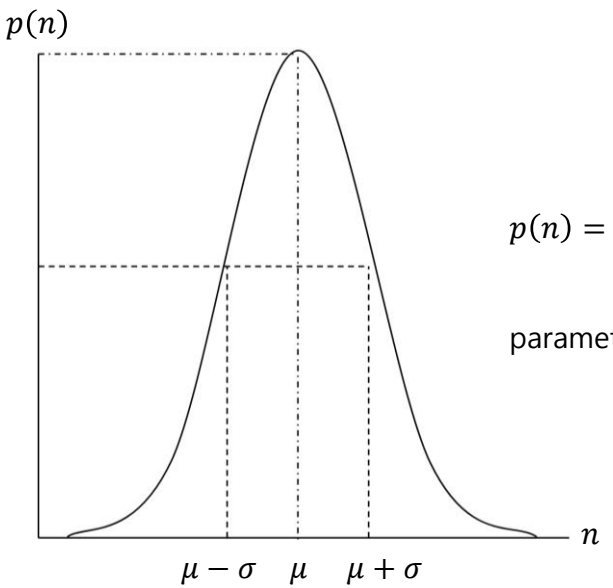
- Random value noise



∗ **degradation $H(x,y)$의 영향이 없다고 가정**

$$\underbrace{I(x,y)}_{\text{손상된 영상}} = \underbrace{O(x,y)}_{\text{원본 영상}} + \underbrace{n(x,y)}_{\text{Noise}}$$

$0 \le x < \mathrm{W}\,,\ 0 \le y < \mathrm{H}$

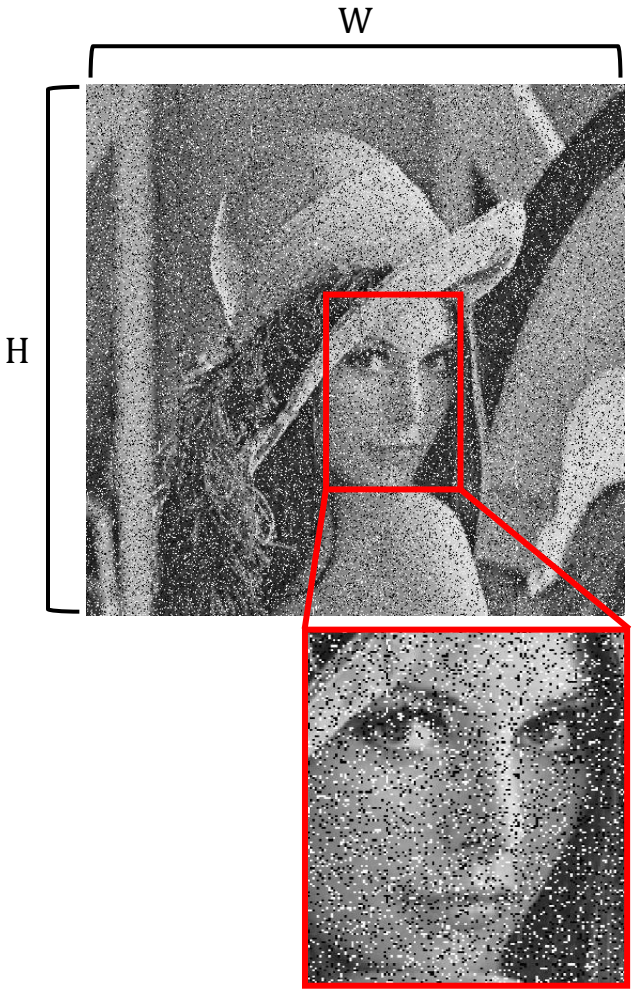


$$p(n) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(n-\mu)^2}{2\sigma^2}}$$

parameter : μ, σ

Salt & pepper noise

- Random position noise



∗ **degradation $H(x,y)$의 영향이 없다고 가정**

$$\underbrace{I(x,y)}_{\text{손상된 영상}} = \underbrace{O(x,y)}_{\text{원본 영상}} + \underbrace{n(x,y)}_{\text{Noise}}$$

- **(전체 픽셀 수 × ρ) 개의 random position**

$$\rightarrow I(x,y) = \underbrace{255}_{\text{maximum value}} \quad \text{or} \quad I(x,y) = \underbrace{0}_{\text{minimum value}}$$

- **Otherwise**

$$\rightarrow I(x,y) = O(x,y)$$

parameter : ρ

(0.0 ~ 1.0)

(0% ~ 100%)

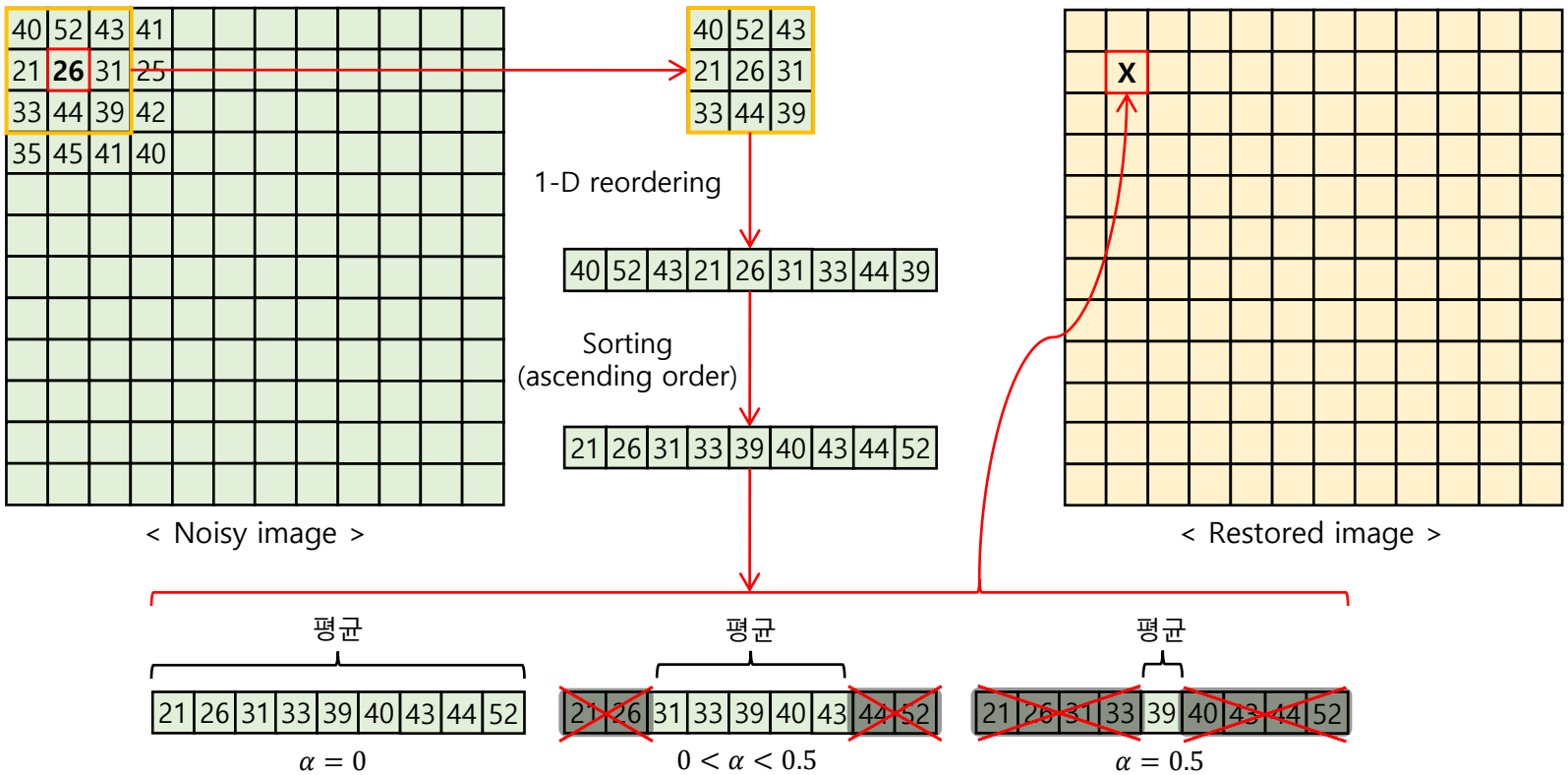
IMAGE RESTORATION

Alpha-trimmed mean filtering

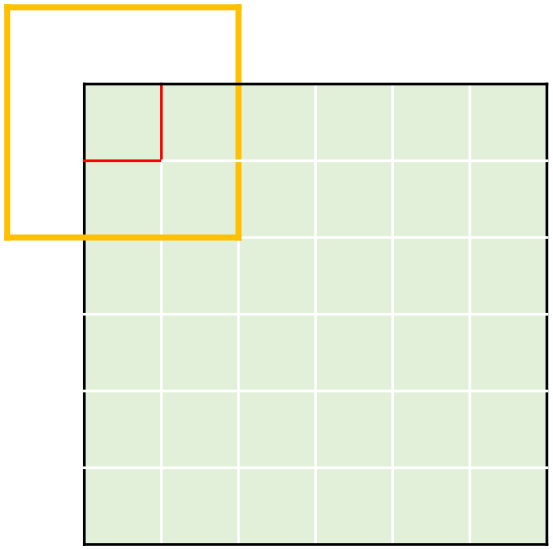
- 중간값 주변의 평균을 필터링 결과로 사용
  - The number of trimmed elements : $\mathrm{floor}(\alpha \times (f \times f))$
  - ex. 3 x 3 filtering

α : parameter

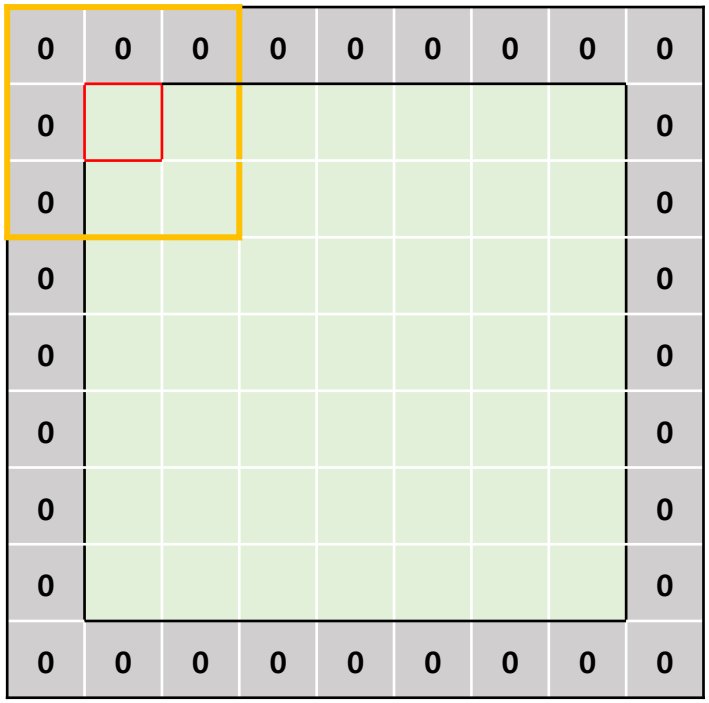
f : filter size



Padding : Zero padding

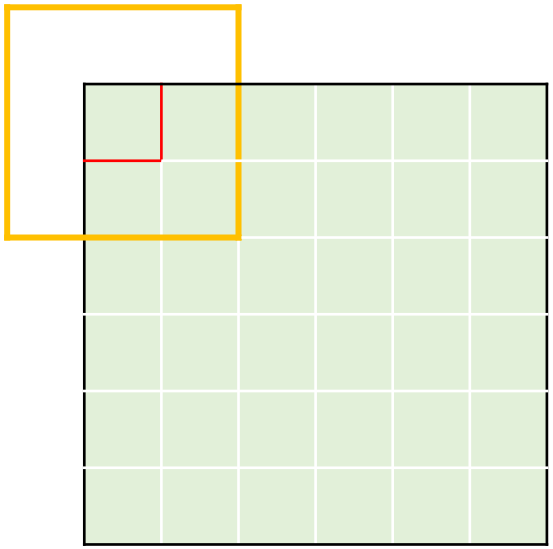


< Noisy image >

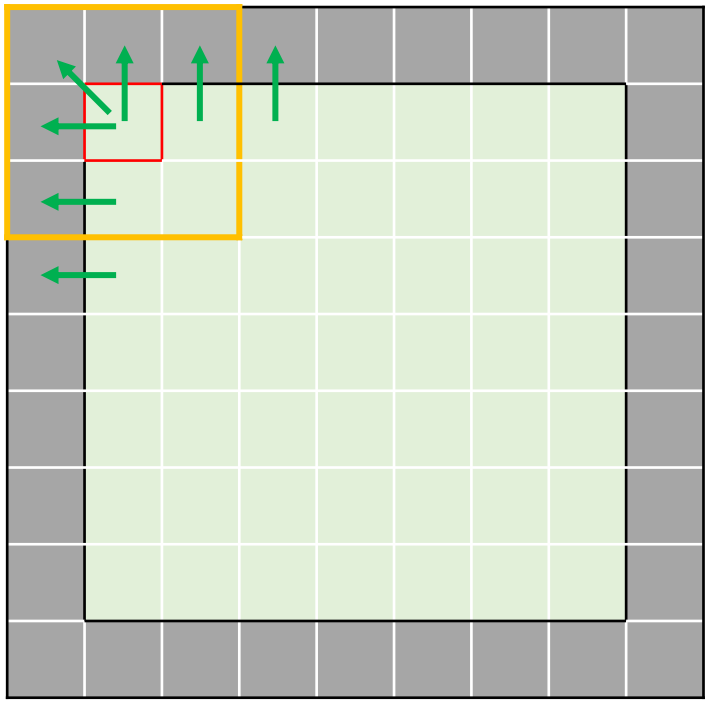


< Noisy & padded image >

Padding : Copy padding



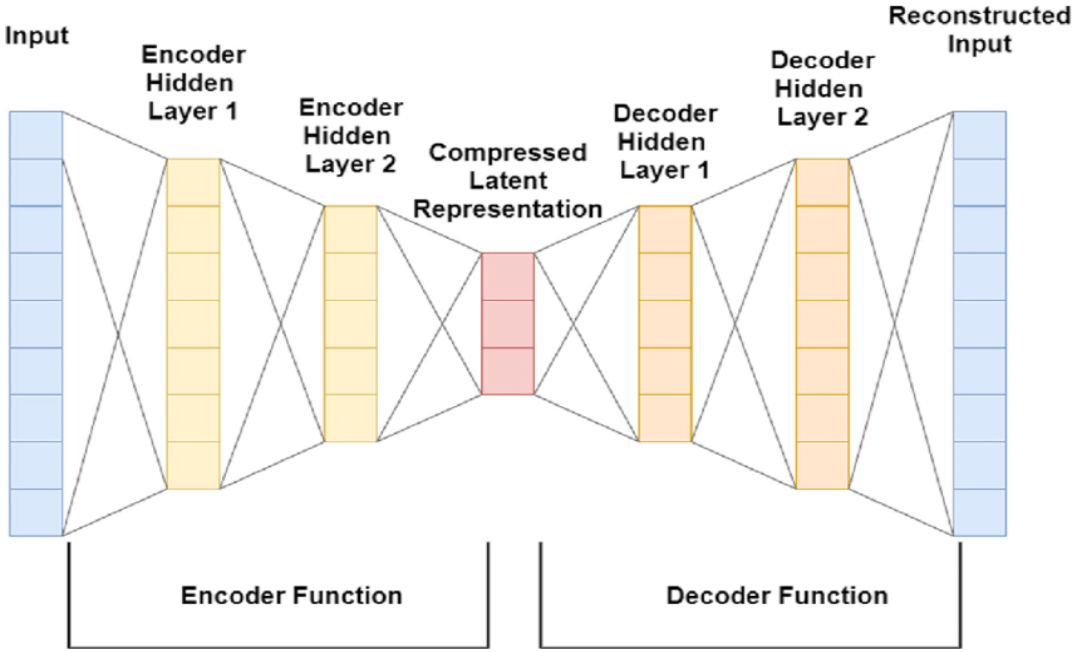
< Noisy image >



< Noisy & padded image >

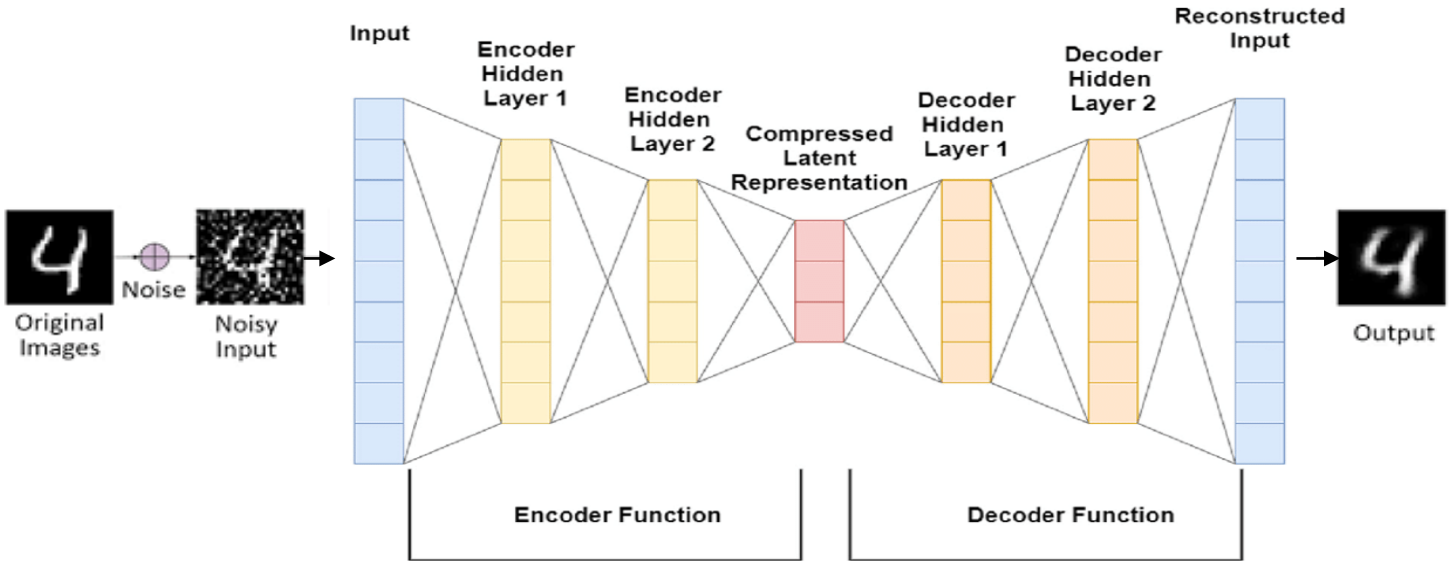
AutoEncoder

- 오토인코더
  - 입력층의 구조와 출력층의 구조가 동일하게 구축된 다층 신경망
    - 출력값이 입력값과 유사해지도록 학습
  - 오토인코더 구조
    - 인코더 (인지 네트워크, Recognition network)
    - 디코더 (생성 네트워크, Generative network)



Denoising AutoEncoder

- 입력 영상에 임의의 노이즈를 추가하여 학습
  - 출력 영상이 노이즈가 추가되지 않은 입력 영상과 유사해지도록 학습



PSNR (Peak Signal to Noise Ratio)

- PSNR이란
  - 최대 신호 대 왜곡 비율을 의미
  - 객관적 화질 평가에서 널리 사용되는 화질 척도

- 손실이 적을수록 높은 PSNR 값을 가짐
  - MSE (Mean Square Error)가 0으로 열화가 없는 경우, PSNR은 무한대

$$MSE = \frac{\sum_{h=0}^{H-1}\sum_{w=0}^{W-1}(I_{ori}(w,h)-I'(w,h))^2}{W\times H}$$

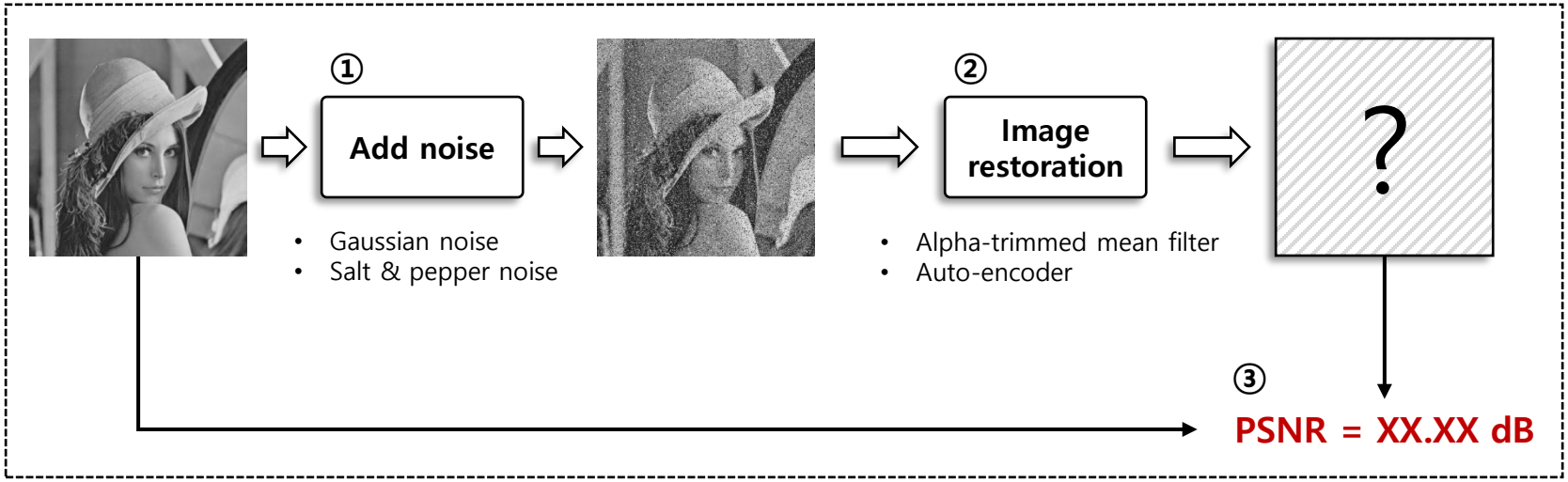
$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right)$$

- MAX : 입력 신호의 최댓값 (8비트 심도일 경우, 255)
  (0 ~ 1 범위로 normalize 된 경우, 1)

과제 세부 사항

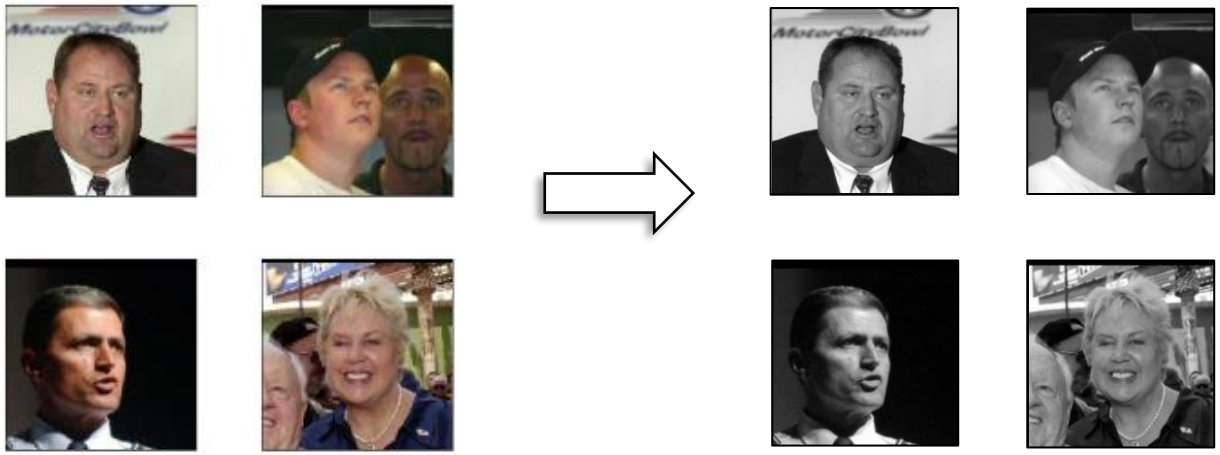
과제 세부 사항

1. 잡음을 생성하여 입력 이미지에 추가
   - Gaussian noise
   - Salt & pepper noise
2. 잡음이 추가된 이미지에 대해 잡음 제거 알고리즘 수행
   - Alpha-trimmed mean filter
   - Auto-encoder
3. PNSR을 측정하여 잡음 제거 성능 비교



과제 세부 사항

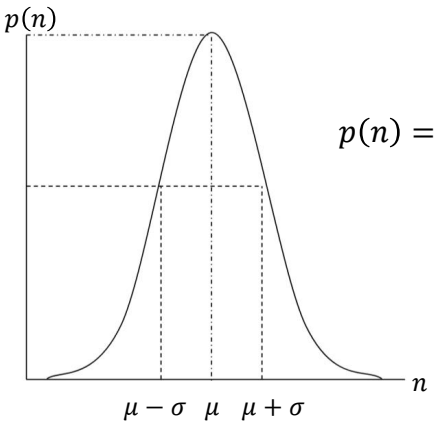
- 데이터셋
  - lfw (Labeled Faces in the Wild) datasets
    - Train set : 13233개
    - 250 x 250 color image
  - **과제 데이터셋**
    - Train set : 3000개, Test set : 100개
    - 256 x 256 grayscale image



* 과제 수행 간 데이터셋 로드 방법은 아래 개발 환경 슬라이드 참고

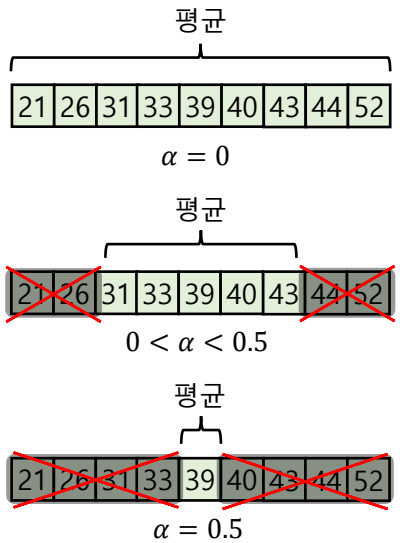
과제 수행 간 파라미터 설정

- Add noise
  - Gaussian noise
    - $\boldsymbol{\mu = 0,\ \sigma = 10}$
  - Salt & pepper noise
    - $\boldsymbol{\rho = 0.2\ (20\%)}$



$$p(n) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(n-\mu)^2}{2\sigma^2}}$$

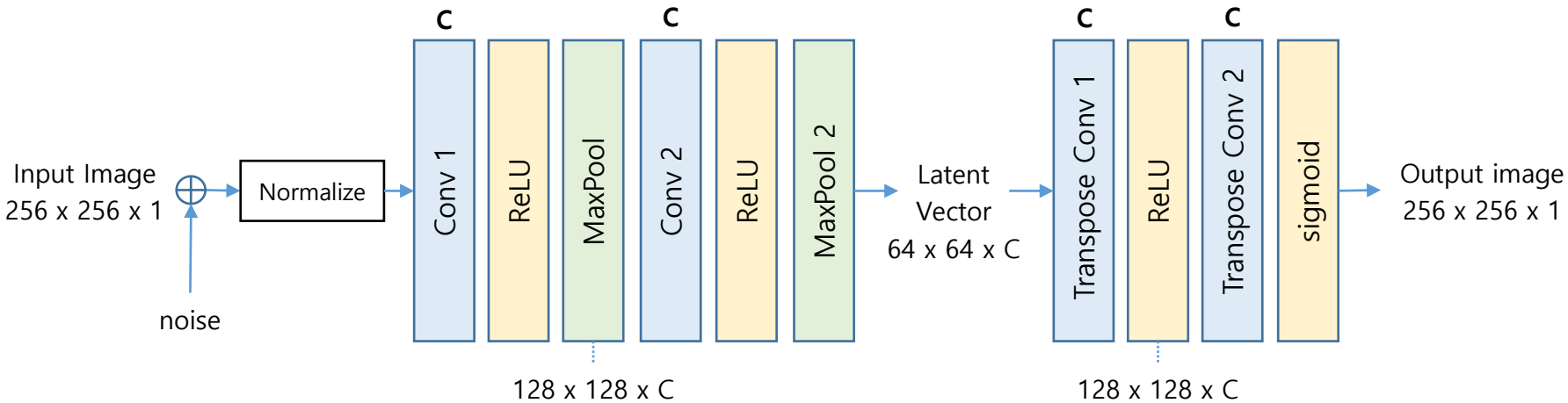
- Image restoration
  - Alpha-trimmed mean filter
    - Setting ①
      - ➢ **Filter size: 3x3, $\boldsymbol{\alpha = 0.2}$**
    - Setting ②
      - ➢ **Filter size: 5x5, $\boldsymbol{\alpha = 0.4}$**



N of trimmed elements :
$\text{floor}(\alpha \times (f \times f))$

과제 수행 간 파라미터 설정

- Image restoration
  - Auto-encoder
    - Conv2D & Conv2DTranspose
      - ➢ Number of channels (**C**): **32 or 64** / Kernel size: (3x3) / Activation: ReLU
    - Maxpooling2D
      - ➢ pool_size: (2x2)
    - 학습 과정
      - ➢ Optimizer: adam
      - ➢ Loss function: MSE (Mean Squared Error)
      - ➢ Epochs: 10



과제 세부 사항

- 2종류의 잡음에 대해 4가지 잡음 제거 방법으로 실험을 진행한 후, PSNR 측정 수행
  - 아래와 같은 표로 정리하여 보고서에 작성
    - test dataset 100장에 대한 PSNR 측정 후 비교

	Alpha-trimmed mean filtering		**Autoencoder**	
Parameter setting	Filter 3x3, $\alpha = 0.2$	Filter 5x5, $\alpha = 0.4$	32 channels	64 channels
Gaussian noise				
Salt & pepper noise				

과제 제출

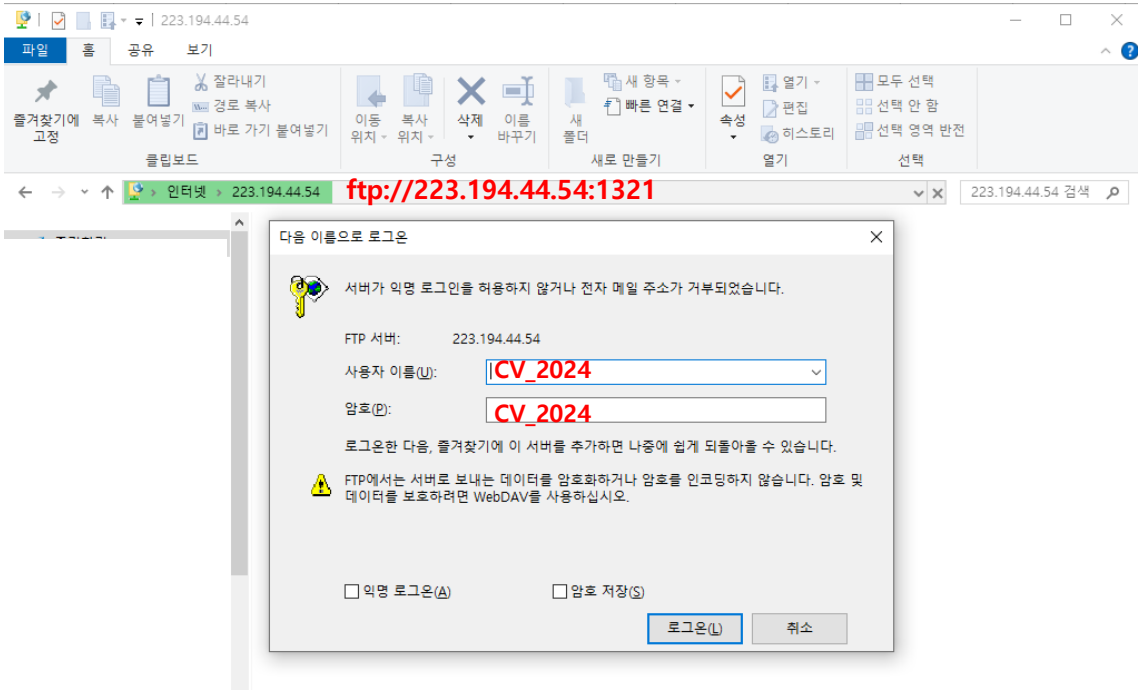
- 제출물 코드, 보고서 압축한 파일 제출 (학번_이름_ver#.zip) (예: 2024123456_홍길동_ver2.zip)
  - 코드
    - 과제 수행한 python 또는 Colab .ipynb 파일
    - 코드에 주석 작성
  - 보고서
    - 과제 개요, 과제 수행 방법, 결과 분석, 고찰
      - ➢ 개발환경 버전 명시 (python, tensorflow, keras & Colab 사용 유무)

- 제출처
  - FTP Sever

- 마감일
  - **2022년 4월 28일 (일요일) 23:59:59** (서버 시간 기준)
  - 마감일 이후 ~ 일주일: 채점 점수의 50%만 실제 과제 점수로 반영
  - 일주일 이후: 0점 처리

과제 제출 방법

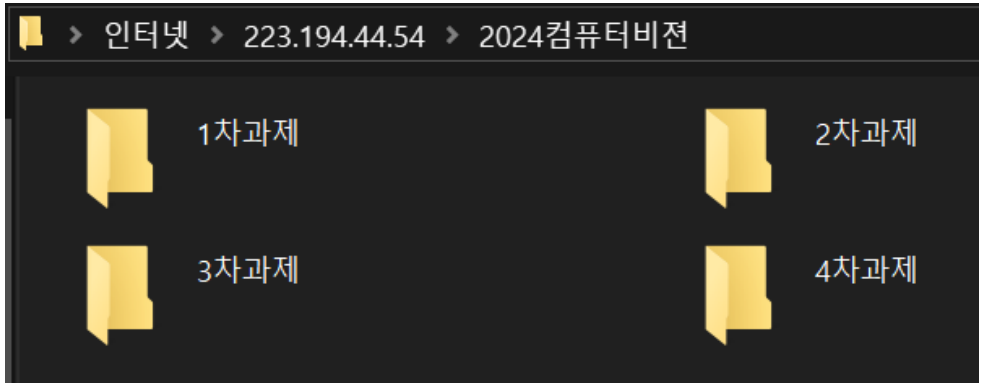
- FTP Server
  - Windows 파일 탐색기 또는 FileZilla 이용해 서버 접속
    - URL: ftp://223.194.44.54:1321
    - Username: CV_2024
    - Password: CV_2024



과제 제출 방법

- FTP Server
  - 과제 폴더에 과제 압축 파일 업로드
    - 업로드 후 파일 삭제 불가능함 → 과제 수정 시, 새로운 버전으로 업로드
    - 마지막 버전의 과제 압축 파일로 과제 채점 예정
      - Ex. 2024123456_홍길동_ver1.zip
        2024123456_홍길동_ver2.zip
        2024123456_홍길동_ver3.zip

      → "2024123456_홍길동_ver3.zip" 파일로 과제 채점

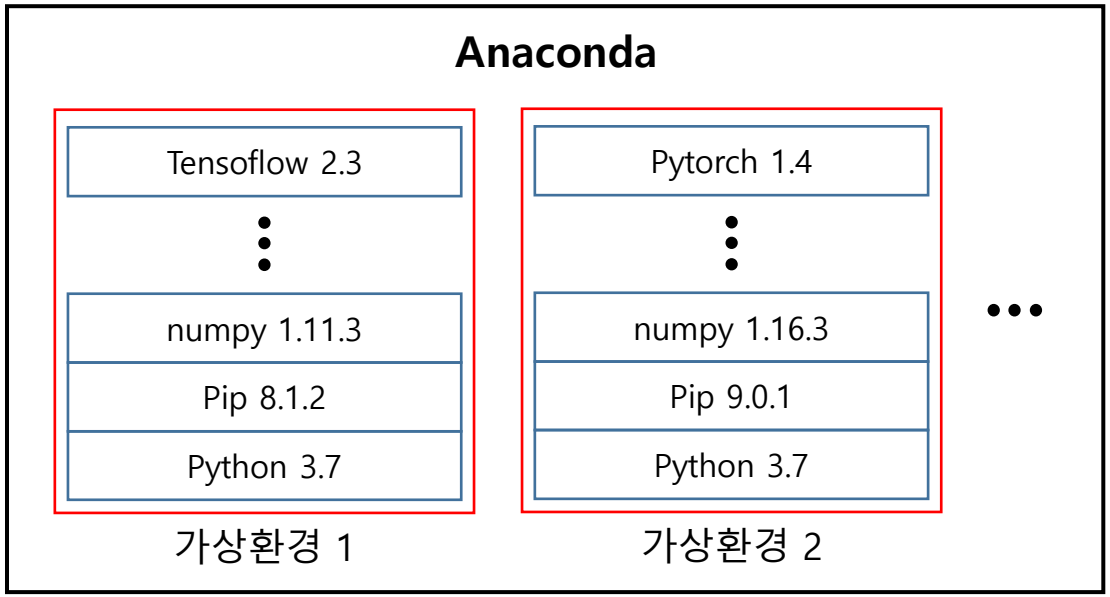


PYTHON 개발 환경

ANACONDA + VISUAL STUDIO CODE

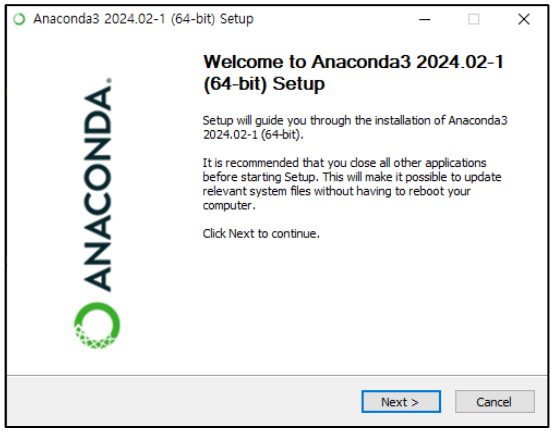
Anaconda

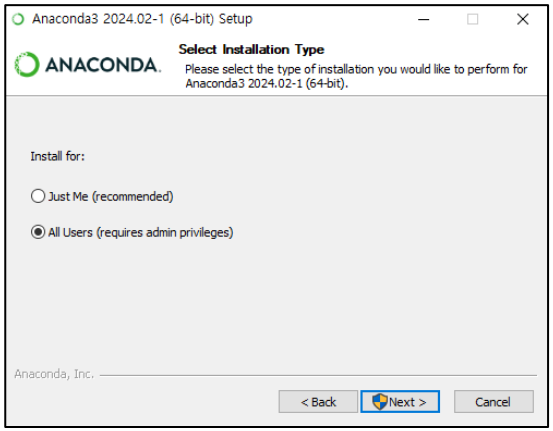
- Anaconda
  - 수학과 과학 분야에서 사용되는 여러 패키지들을 묶어 놓은 패키지 및 환경 관리 소프트웨어
    - SciPy, Numpy, Matplotlib, Pandas, Tensorflow 등을 비롯한 많은 패키지들을 포함
  - 가상환경을 제공하여 버전 및 패키지 관리 용이

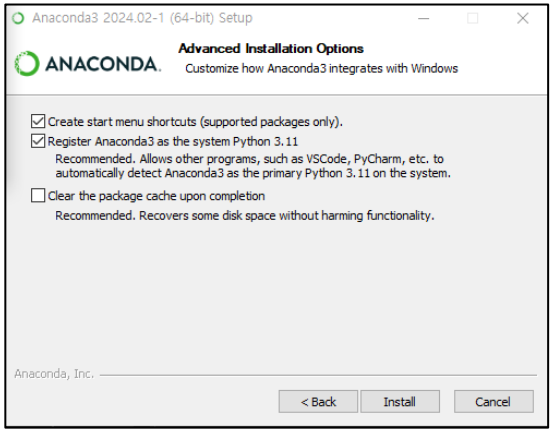


Anaconda

- Anaconda 설치
  - 설치 파일 다운로드 후 설치 수행
    - https://www.anaconda.com/download#downloads

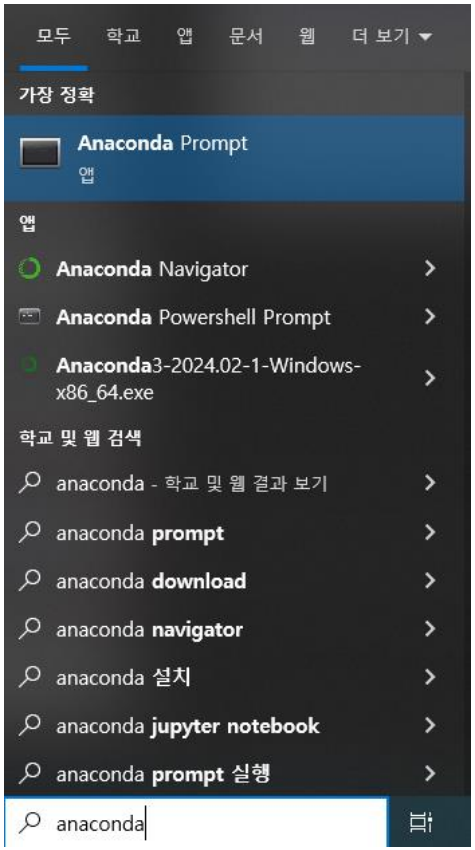


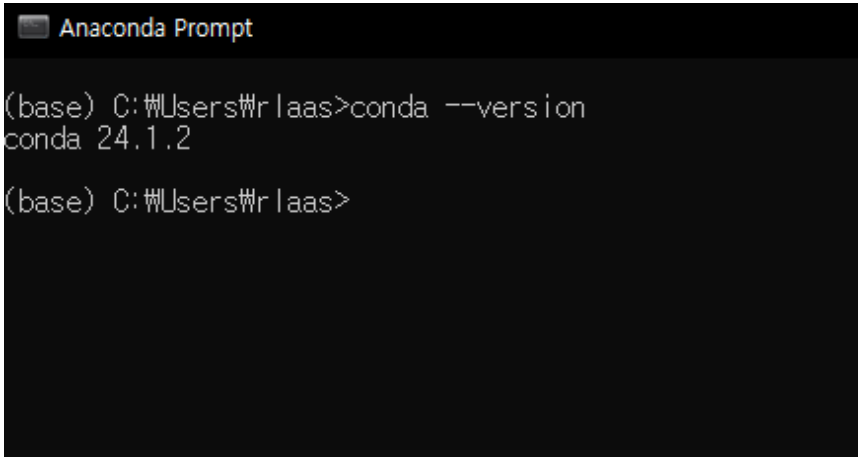




Anaconda

- Anaconda 실행
  - 윈도우 시작 메뉴에서 Anaconda Prompt 실행





Anaconda

- Commands
  - 가상환경 생성
    - conda create --name [생성할 가상환경 이름] python=버전명
      - ➢ ex. conda create --name CV2024 python=3.10

```
(base) C:₩Users₩rlaas>conda create --name CV2024 python=3.10
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##
```

  - 생성된 가상환경 목록 확인
    - conda info --envs

```
(base) C:₩Users₩rlaas>conda info --envs
# conda environments:
#
base                  *  C:₩ProgramData₩anaconda3
CV2024                   C:₩Users₩rlaas₩.conda₩envs₩CV2024
```

해당 가상환경의 interpreter
및 패키지가 위치하는 장소

Anaconda

- Commands
  - 생성한 가상환경으로 활성화 및 비활성화
    - activate [가상환경이름]

```
(base) C:₩Users₩rlaas>activate CV2024

(CV2024) C:₩Users₩rlaas>
```

    - deactivate

```
(CV2024) C:₩Users₩rlaas>conda deactivate

(base) C:₩Users₩rlaas>
```

  - 가상환경 내에 설치된 패키지 목록확인
    - conda list

```
(CV2024) C:₩Users₩rlaas>conda list
# packages in environment at C:₩Users₩rlaas₩.conda₩envs₩CV2024:
#
# Name                    Version                   Build  Channel
bzip2                     1.0.8                h2bbff1b_5
ca-certificates           2024.3.11            haa95532_0
libffi                    3.4.4                hd77b12b_0
openssl                   3.0.13               h2bbff1b_0
pip                       23.3.1          py310haa95532_0
python                    3.10.14              he1021f5_0
setuptools                68.2.2          py310haa95532_0
sqlite                    3.41.2               h2bbff1b_0
tk                        8.6.12               h2bbff1b_0
tzdata                    2024a                h04d1e81_0
vc                        14.2                 h21ff451_1
vs2015_runtime            14.27.29016          h5e58377_2
wheel                     0.41.2          py310haa95532_0
xz                        5.4.6                h8cc25b3_0
zlib                      1.2.13               h8cc25b3_0
```

Anaconda

- Commands
  - 기존 가상환경 복제
    - conda create --name [새로 생성할 가상환경 이름] --clone [복제할 기존 가상환경 이름]
    - 기존 가상환경의 패키지를 복사하여 새로운 가상환경 생성

```
(CV2024) C:₩Users₩rlaas>conda create --name test --clone CV2024
Source:      C:₩Users₩rlaas₩.conda₩envs₩CV2024
Destination: C:₩Users₩rlaas₩.conda₩envs₩test
Packages: 15
Files: 1

Downloading and Extracting Packages:


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate test
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(CV2024) C:₩Users₩rlaas>conda info --envs
# conda environments:
#
base                     C:₩ProgramData₩anaconda3
CV2024                *  C:₩Users₩rlaas₩.conda₩envs₩CV2024
test                     C:₩Users₩rlaas₩.conda₩envs₩test
```

  - 가상환경 지우기
    - conda env remove -n [가상환경 이름]

```
(CV2024) C:₩Users₩rlaas>conda env remove -n test

Remove all packages in environment C:₩Users₩rlaas₩.conda₩envs₩test:

Everything found within the environment (C:₩Users₩rlaas₩.conda₩envs₩test)
, including any conda environment configurations and any non-conda files,
 will be deleted. Do you wish to continue?
 (y/[n])? y


(CV2024) C:₩Users₩rlaas>conda info --envs
# conda environments:
#
base                     C:₩ProgramData₩anaconda3
CV2024                *  C:₩Users₩rlaas₩.conda₩envs₩CV2024
```

Anaconda

- Commands
  - 아나콘다 버전 확인
    - conda info
    - conda 버전에 따라서 다운가능한 패키지의 버전이 다름

```
(CV2024) C:₩Users₩rlaas>conda info

     active environment : CV2024
    active env location : C:₩Users₩rlaas₩.conda₩envs₩CV2024
            shell level : 2
       user config file : C:₩Users₩rlaas₩.condarc
 populated config files : C:₩Users₩rlaas₩.condarc
          conda version : 24.1.2
    conda-build version : 24.1.2
         python version : 3.11.7.final.0
                solver : libmamba (default)
       virtual packages : __archspec=1=x86_64
                          __conda=24.1.2=0
                          __cuda=12.4=0
                          __win=0=0
       base environment : C:₩ProgramData₩anaconda3  (read only)
      conda av data dir : C:₩ProgramData₩anaconda3₩etc₩conda
  conda av metadata url : None
          channel URLs : https://repo.anaconda.com/pkgs/main/win-64
                         https://repo.anaconda.com/pkgs/main/noarch
                         https://repo.anaconda.com/pkgs/r/win-64
                         https://repo.anaconda.com/pkgs/r/noarch
                         https://repo.anaconda.com/pkgs/msys2/win-64
                         https://repo.anaconda.com/pkgs/msys2/noarch
         package cache : C:₩ProgramData₩anaconda3₩pkgs
                         C:₩Users₩rlaas₩.conda₩pkgs
                         C:₩Users₩rlaas₩AppData₩Local₩conda₩conda₩pkgs
       envs directories : C:₩Users₩rlaas₩.conda₩envs
                         C:₩ProgramData₩anaconda3₩envs
                         C:₩Users₩rlaas₩AppData₩Local₩conda₩conda₩envs
                platform : win-64
```

Anaconda

- Commands
  - 다운가능한 라이브러리 버전 확인
    - conda search 라이브러리명

```
(CV2024) C:₩Users₩rlaas>conda search keras-gpu
Loading channels: done
# Name                       Version           Build  Channel
keras-gpu                      2.0.8  py35hfd8c95c_0  pkgs/main
keras-gpu                      2.0.8  py36hb5f7954_0  pkgs/main
keras-gpu                      2.1.2          py35_0  pkgs/main
keras-gpu                      2.1.2          py36_0  pkgs/main
keras-gpu                      2.1.3          py35_0  pkgs/main
keras-gpu                      2.1.3          py36_0  pkgs/main
keras-gpu                      2.1.4          py35_0  pkgs/main
keras-gpu                      2.1.4          py36_0  pkgs/main
keras-gpu                      2.1.5          py35_0  pkgs/main
keras-gpu                      2.1.5          py36_0  pkgs/main
keras-gpu                      2.1.6          py35_0  pkgs/main
keras-gpu                      2.1.6          py36_0  pkgs/main
keras-gpu                      2.2.0               0  pkgs/main
keras-gpu                      2.2.2               0  pkgs/main
keras-gpu                      2.2.4               0  pkgs/main
keras-gpu                      2.3.1               0  pkgs/main
keras-gpu                      2.4.3               0  pkgs/main
keras-gpu                      2.4.3      hd3eb1b0_0  pkgs/main
keras-gpu                      2.6.0      hd3eb1b0_0  pkgs/main
```

Anaconda

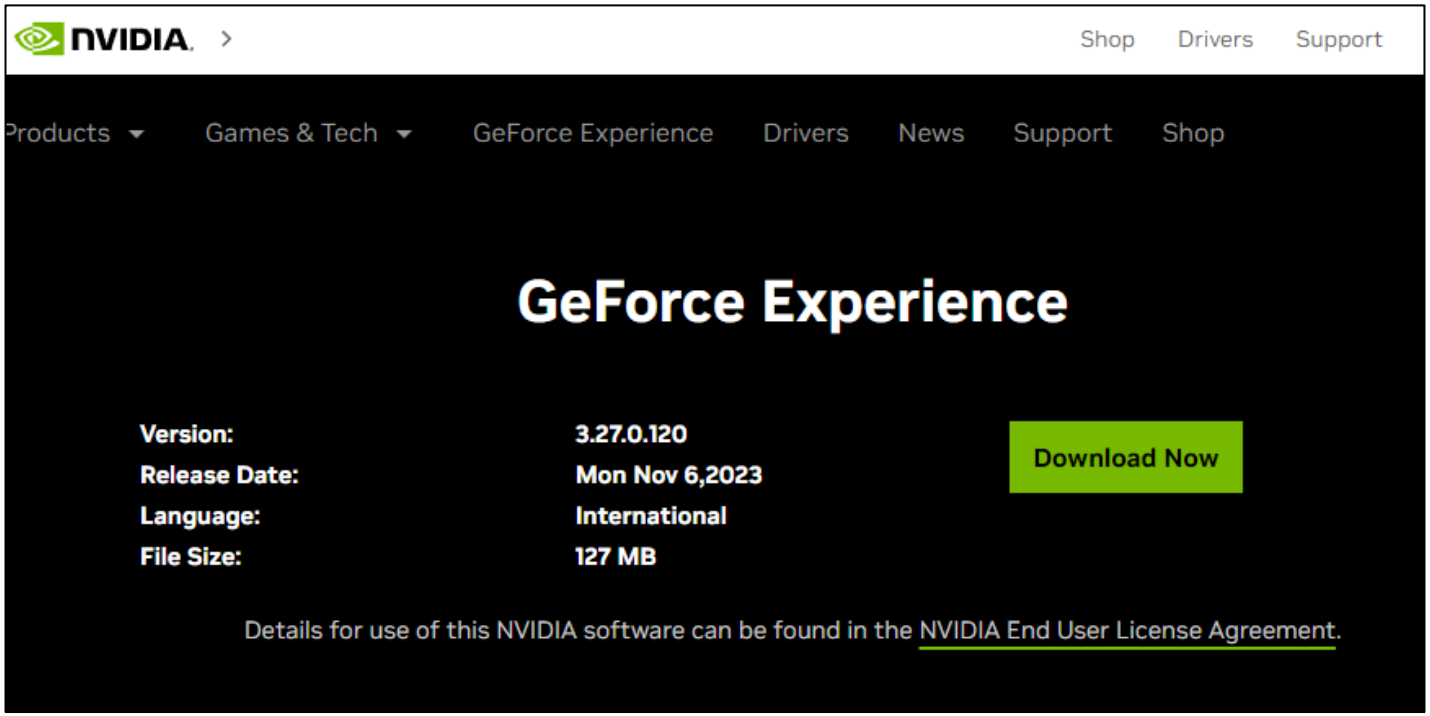
- Commands
  - 라이브러리 설치
    - pip install [라이브러리이름]
    - conda install [라이브러리이름]
      - ➢ 라이브러리 버전을 명시하지 않는 경우 최신버전으로 설치됨

    - pip install [라이브러리이름]==버전
    - conda install [라이브러리이름]==버전
      - ➢ 라이브러리 버전에 맞게 설치 가능

  - 설치된 라이브러리 삭제
    - pip uninstall [삭제할 라이브러리이름]
    - conda uninstall [삭제할 라이브러리이름]

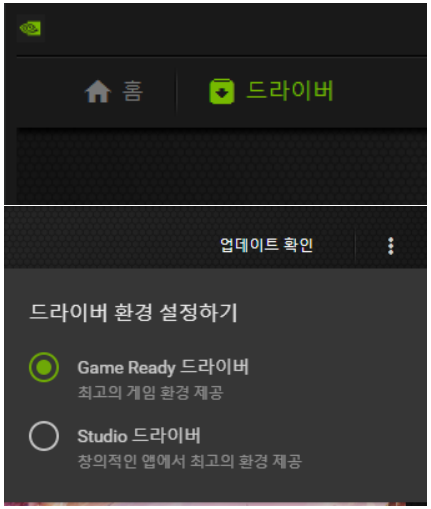
Anaconda

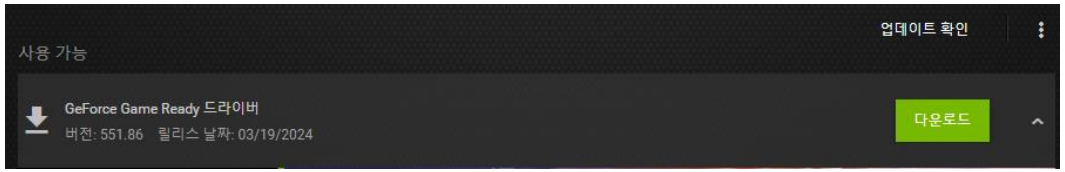
- Tensorflow GPU 설정
  - Geforce Experience 다운로드 후 설치
    - https://www.nvidia.com/en-us/geforce/geforce-experience/download/



Anaconda

- Tensorflow GPU 설정
  - NVIDIA 드라이버 설치
    - Geforce Experience 실행 후 로그인 → 드라이버 탭
    - 우측 점 선택 → Game Ready 드라이버 선택
    - Geforce Game Ready 드라이버 다운로드



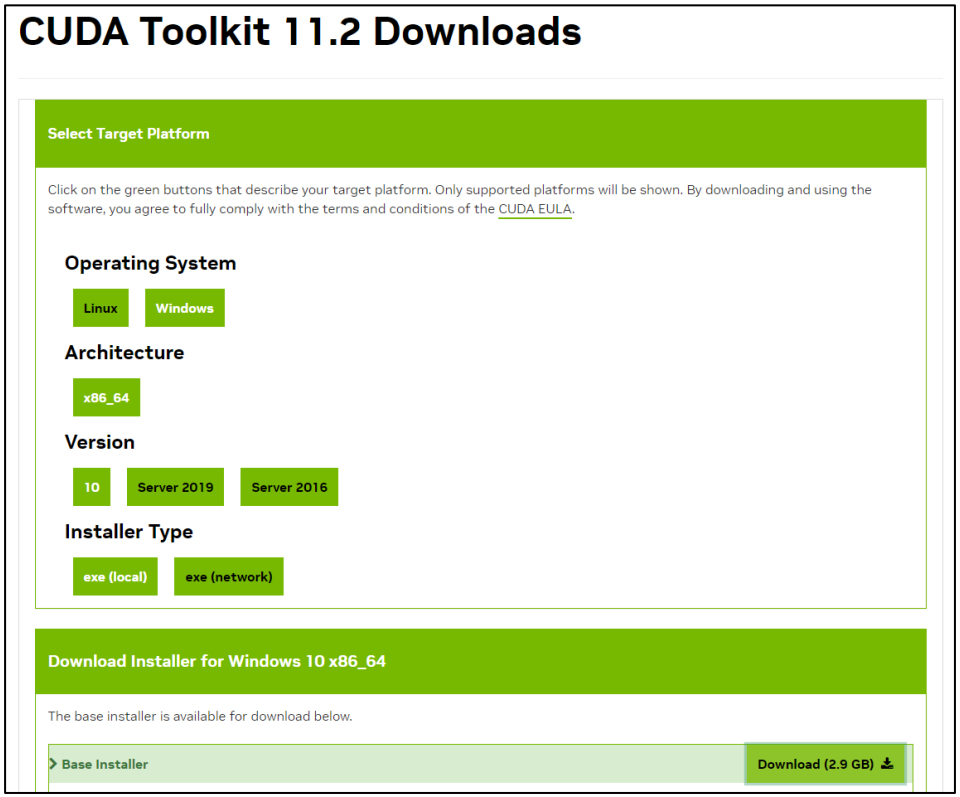


    - 명령어 창에 nvidia-smi 입력하여 GPU 확인

```
(CV2024) C:₩Users₩rlaas>nvidia-smi
Sun Mar 31 21:20:26 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 551.76                 Driver Version: 551.76         CUDA Version: 12.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce RTX 3060      WDDM  |   00000000:07:00.0  On |                  N/A |
|  0%   44C    P8             10W /  170W |    2767MiB /  12288MiB |      3%      Default |
|                                         |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

Anaconda

- Tensorflow GPU 설정
  - CUDA Toolkit 11.2 다운로드 후 설치
    - https://developer.nvidia.com/cuda-11.2.0-download-archive



Anaconda

- Tensorflow GPU 설정
  - cuDNN 8.1 다운로드 후 압축해제
    - https://developer.nvidia.com/rdp/cudnn-archive

Download cuDNN v8.1.0 (January 26th, 2021), for CUDA 11.0,11.1 and 11.2

Library for Windows and Linux, Ubuntu(x86_64, armsbsa, PPC architecture)

cuDNN Library for Linux (aarch64sbsa)

cuDNN Library for Linux (x86_64)

cuDNN Library for Linux (PPC)

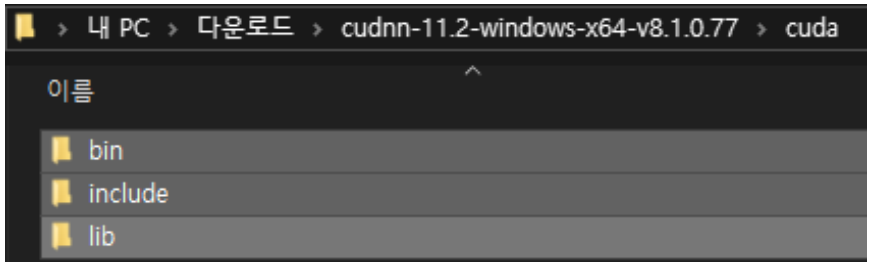
cuDNN Library for Windows (x86)

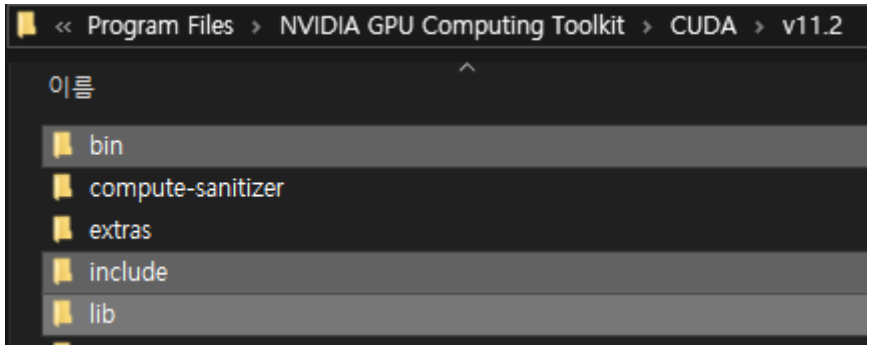
cuDNN Runtime Library for Ubuntu20.04 x86_64 (Deb)

cuDNN Developer Library for Ubuntu20.04 x86_64 (Deb)

Anaconda

- Tensorflow GPU 설정
  - cuDNN 압축해제한 폴더의 bin, include, lib 폴더 아래 경로로 덮어쓰기
    - C:₩Program Files₩NVIDIA GPU Computing Toolkit₩CUDA₩v11.2



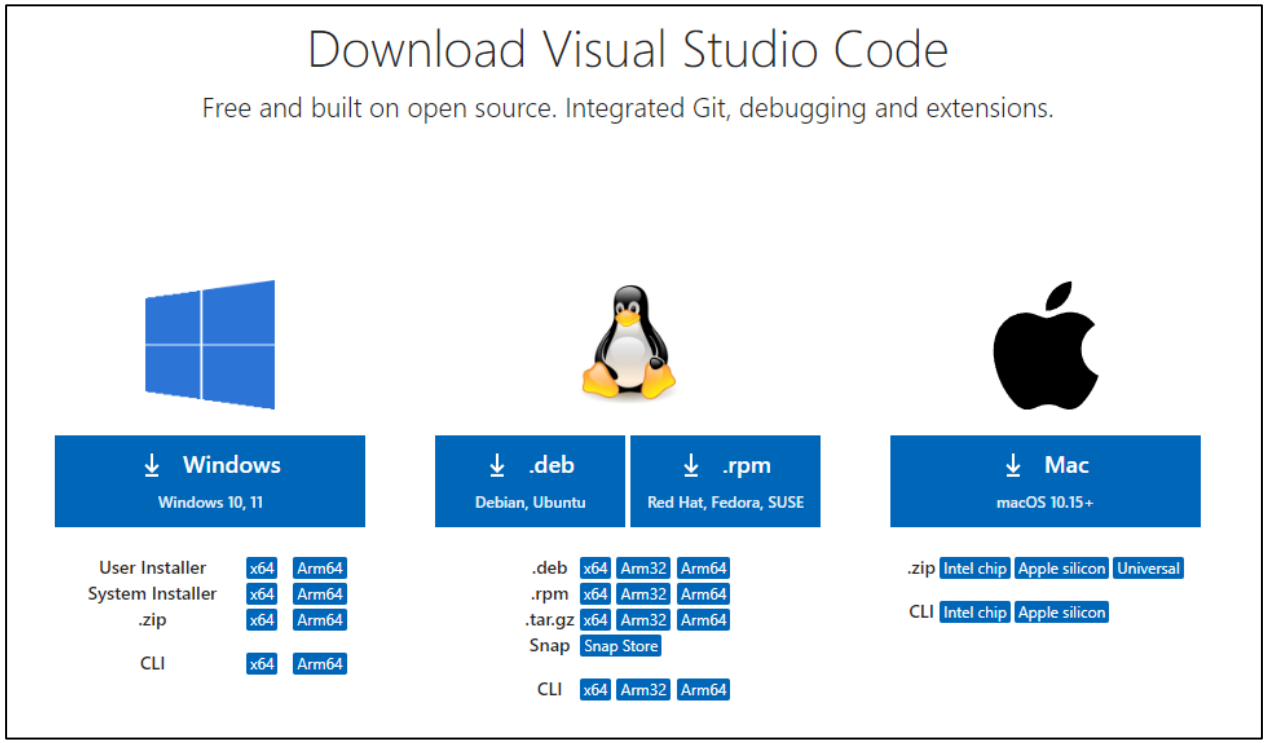


Anaconda

- 라이브러리 설치
  - tensorflow
    - pip install tensorflow==2.10.0
    - pip install tensorflow-gpu==2.10.0
  - matplotlib
    - pip install matplotlib
  - scikit-image
    - pip install scikit-image

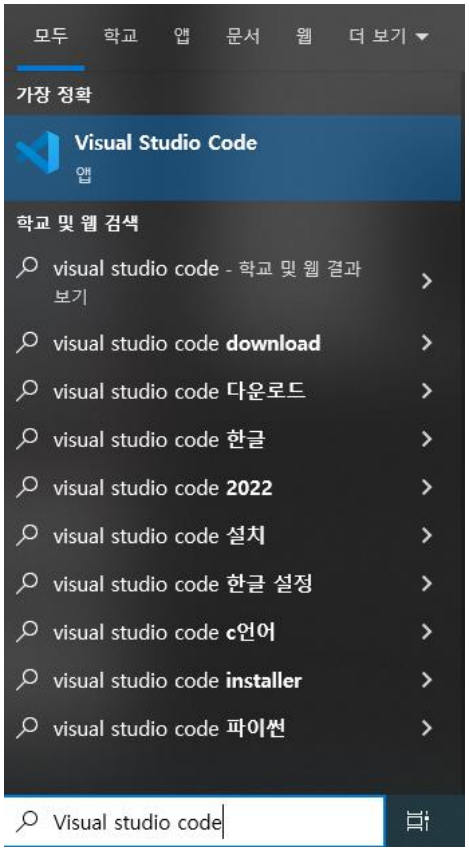
Visual Studio Code

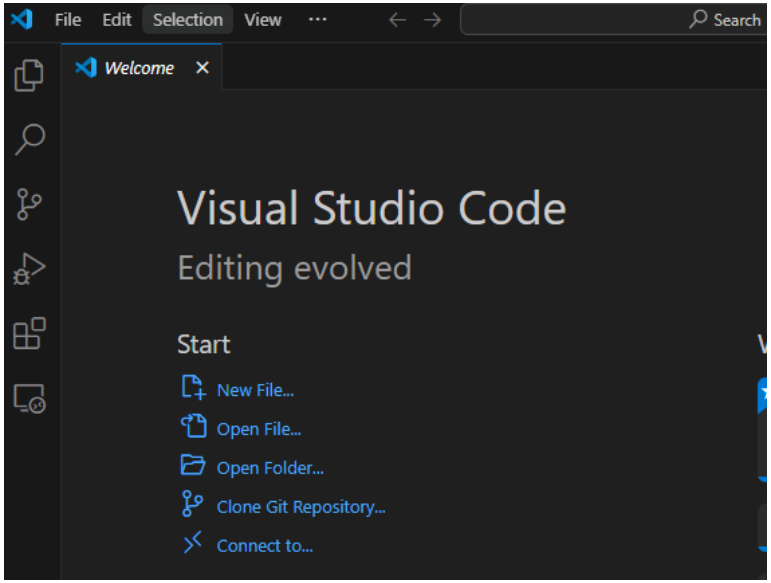
- Visual Studio Code 설치
  - 설치 파일 다운로드 후 설치 수행
    - https://code.visualstudio.com/download



Visual Studio Code

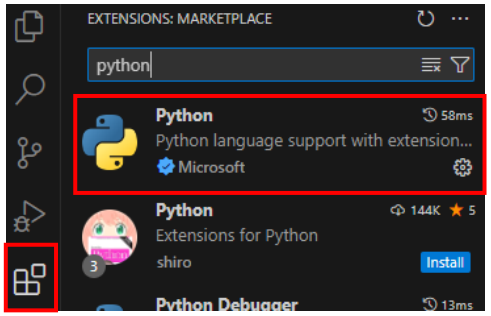
- Visual Studio Code 실행
  - 윈도우 시작 메뉴에서 Visual Studio Code 실행



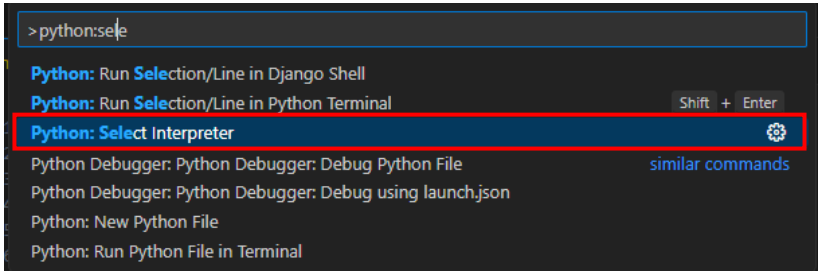


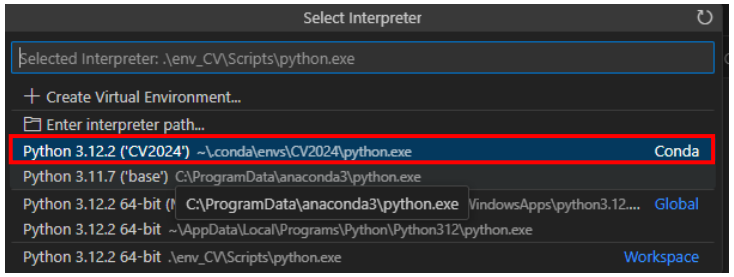
Visual Studio Code

- Python & Anaconda 연동
  - File → Open Folder → [개발 수행 폴더 선택]
  - 좌측 Extensions 아이콘 선택 → "Python" extension 설치



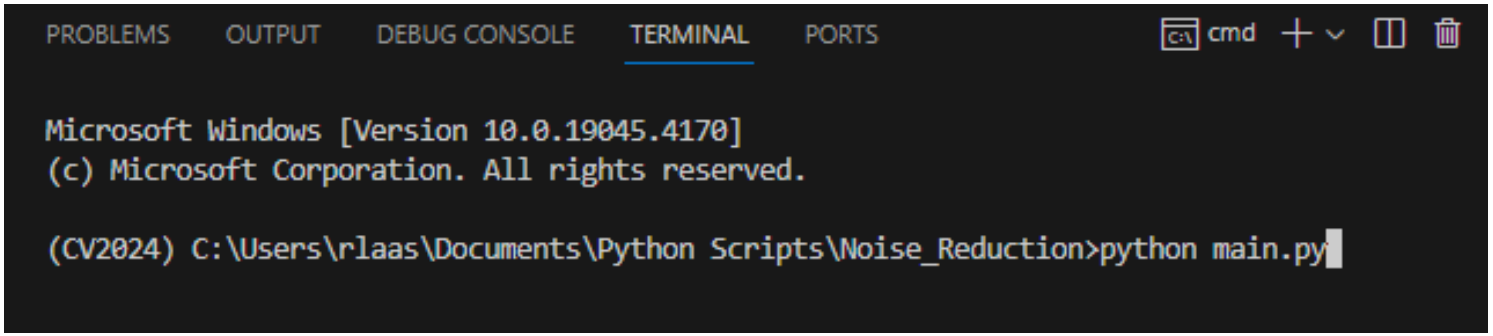
  - ctrl + shift + p → "Python: Select interpreter" 선택
    - 생성한 anaconda 가상 환경 선택





Visual Studio Code

- Python & Anaconda 연동
  - ctrl + `
    - cmd (Command Prompt) 터미널 통해 가상환경 확인
  - python [파이썬 파일]
    - cmd (Command Prompt) 터미널 통해 코드 실행 가능



Visual Studio Code

- Tensorflow GPU 설정 확인

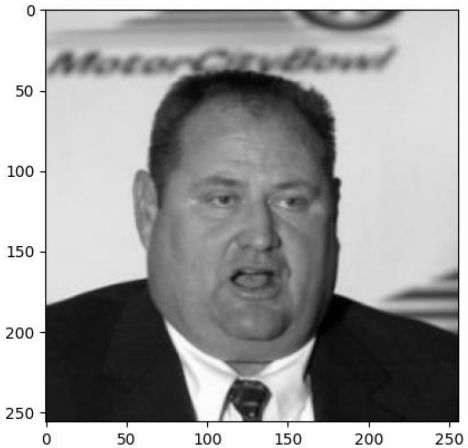
```
(CV2024) C:\Users\rlaas\Documents\Python Scripts\Noise_Reduction>python
Python 3.10.14 | packaged by Anaconda, Inc. | (main, Mar 21 2024, 16:20:14) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from tensorflow.python.client import device_lib
>>> device_lib.list_local_devices()
2024-03-31 23:54:01.834033: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-03-31 23:54:02.279012: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device /device:GPU:0 wit
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 17285284039143292348
xla_global_id: -1
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 10057940992
locality {
  bus_id: 1
  links {
  }
}
incarnation: 12057095033962505142
physical_device_desc: "device: 0, name: NVIDIA GeForce RTX 3060, pci bus id: 0000:07:00.0, compute capability: 8.6"
xla_global_id: 416903419
]
>>> quit()

(CV2024) C:\Users\rlaas\Documents\Python Scripts\Noise_Reduction>
```

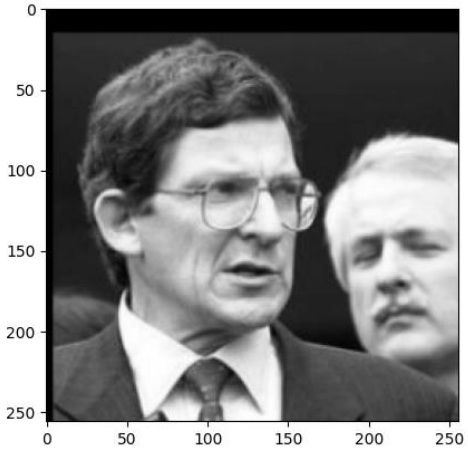
Anaconda + Visual Studio Code

- 데이터셋 로드 방법
  - 제공된 dataset.npy 파일 개발 경로에 저장 후 다음 코드를 통해 로드

```python
import math
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt


if __name__ == '__main__':

  dataset_resize  = np.load('dataset.npy')
  train_data      = dataset_resize[:3000]
  test_data       = dataset_resize[3000:]
  data_shape      = train_data[0].shape

  print(train_data.shape)
  print(test_data.shape)

  plt.imshow(train_data[0], cmap='gray')
  plt.show()
  plt.imshow(test_data[0], cmap='gray')
  plt.show()
```

```
(3000, 256, 256)
(100, 256, 256)
```



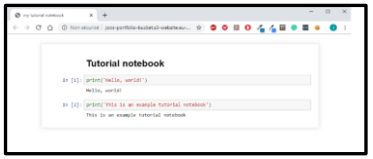


PYTHON 개발 환경

GOOGLE COLAP

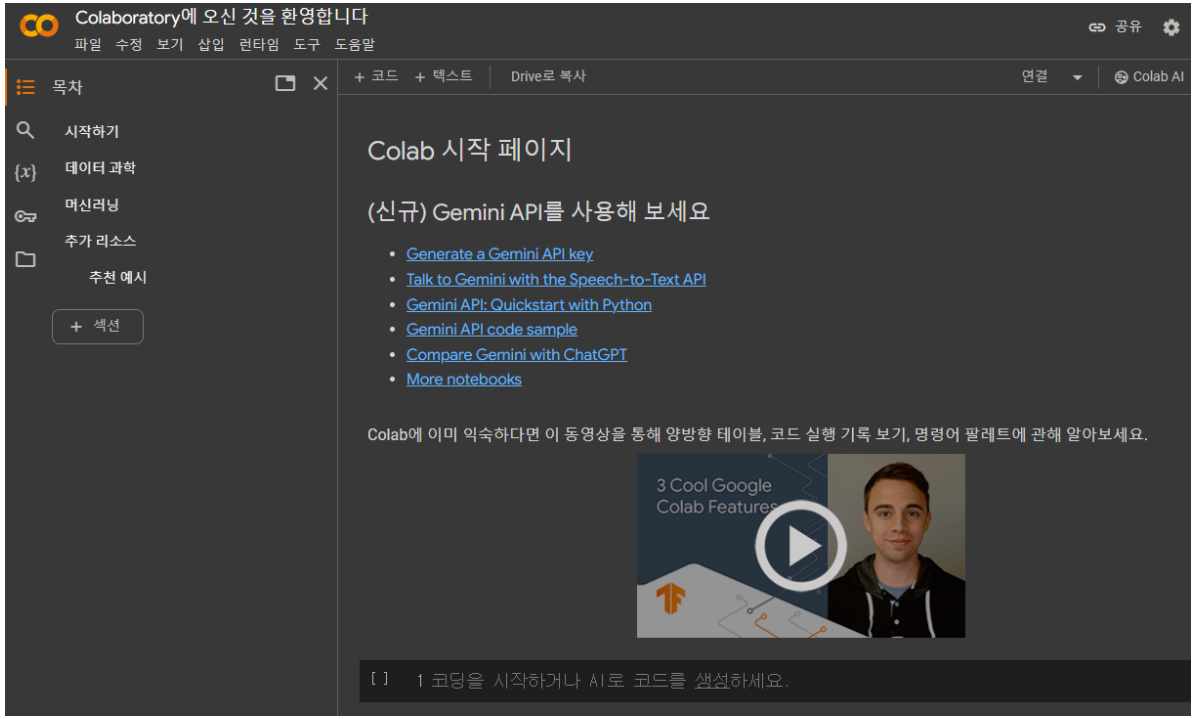
Google Colab

- Google Colaboratory (Colab)
  - 별도의 파이썬 설치 없이 웹 브라우저만을 이용해 주피터 노트북과 같은 작업가능
  - 주피터 노트북을 구글 서버에서 가동시키고 사용자가 조작
  - Google Drive처럼 협업 가능
  - Pytorch, Tensorflow, keras, matplotlib, scikit-learn, pandas 등의 라이브러리가 미리 설치되어 있음
- 구글 gmail 계정을 통해 서비스 사용 가능
- 무료로 GPU 사용가능
  - 무료로 GPU 사용시 하루 최대 12시간 (사용량에 따라 감소함)



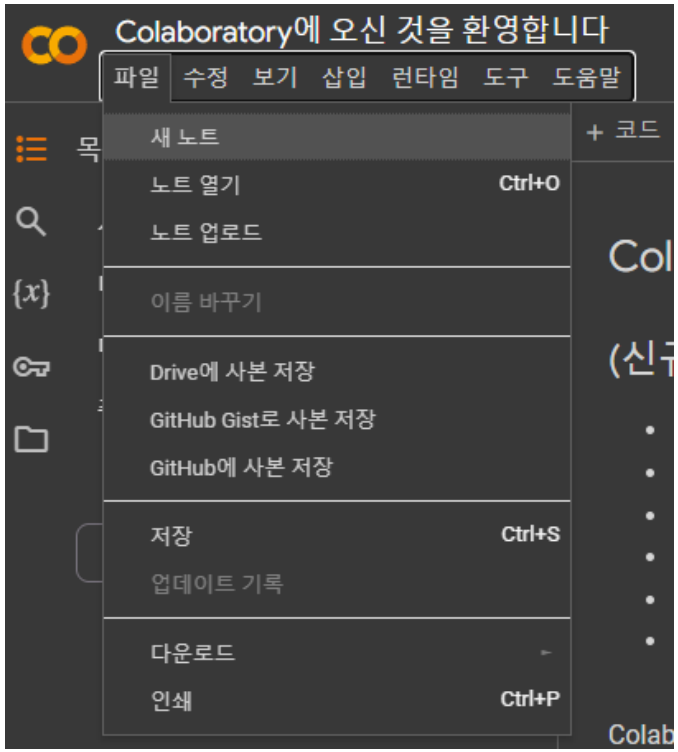
Google Colab

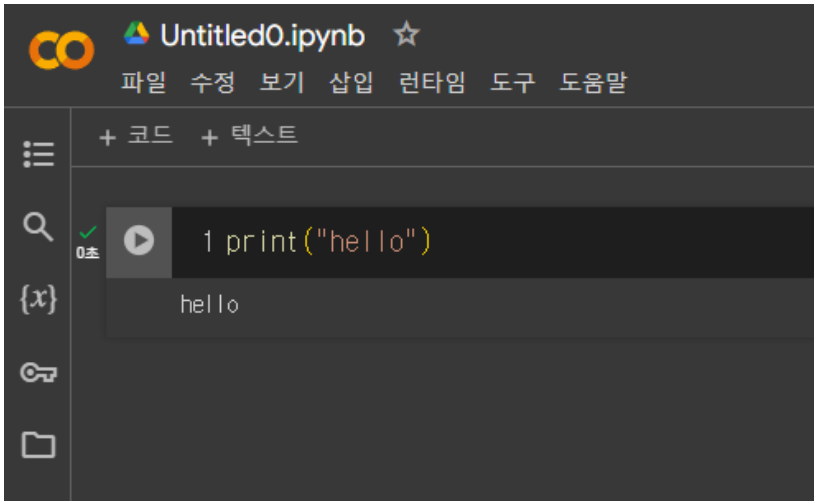
- G-mail 계정 생성 및 로그인
  - https://accounts.google.com/ServiceLogin
- Colab 사이트 접속
  - https://colab.research.google.com/



Google Colab

- 새 노트북 생성
  - 파일 → 새 노트 선택
  - 셀 안에 파이썬 코드를 입력하여 실행





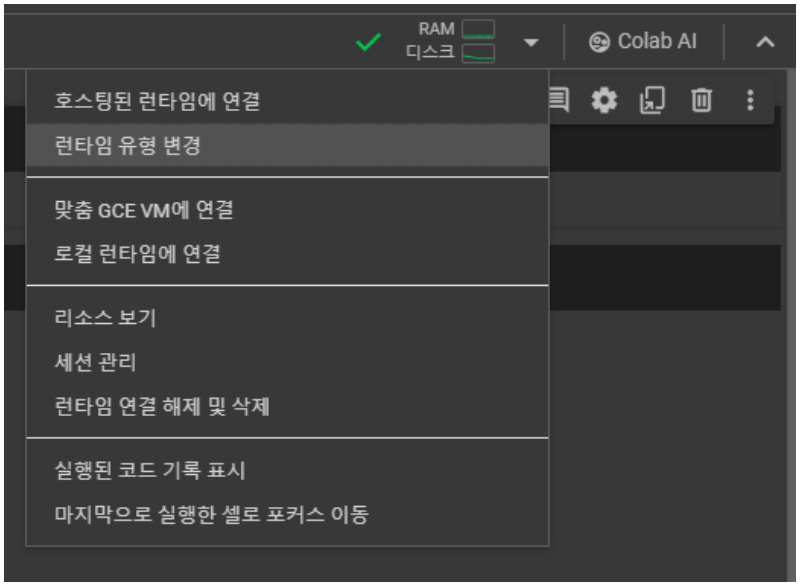
Google Colab

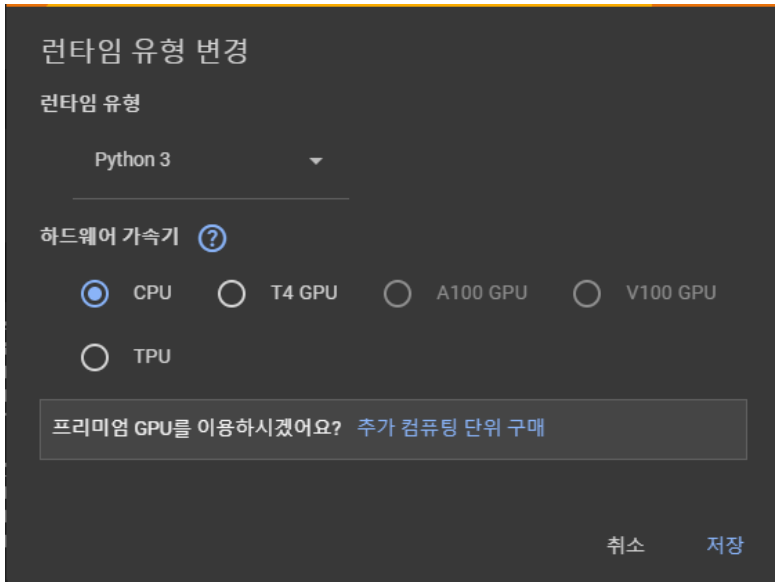
- 운영체제 및 하드웨어 사양 확인하기

```
1 !cat /etc/issue.net # 운영체제 사양

Ubuntu 22.04.3 LTS

[3] 1 !head /proc/cpuinfo # CPU 사양

processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0xffffffff
cpu MHz         : 2199.998
cache size      : 56320 KB
physical id     : 0

[5] 1 !head -n 3 /proc/meminfo # 메모리 사양

MemTotal:       13290480 kB
MemFree:         8648712 kB
MemAvailable:   12343100 kB

[6] 1 !df -h # 디스크 사양

Filesystem      Size  Used Avail Use% Mounted on
overlay         108G   25G   84G  23% /
tmpfs            64M    0   64M   0% /dev
shm             5.8G    0  5.8G   0% /dev/shm
/dev/root       2.0G  1.1G  849M  57% /usr/sbin/docker-init
tmpfs           6.4G  256K  6.4G   1% /var/colab
/dev/sda1        70G   43G   27G  62% /kaggle/input
tmpfs           6.4G    0  6.4G   0% /proc/acpi
tmpfs           6.4G    0  6.4G   0% /proc/scsi
tmpfs           6.4G    0  6.4G   0% /sys/firmware
```

Google Colab

- GPU 사용하기
  - 런타임 → 런타임 유형 변경 → 하드웨어 가속기 설정 → T4 GPU로 변경 (TPU로도 변경가능)





Google Colab

- GPU 사양 확인하기
  - !nvidia-smi

```
1 !nvidia-smi

Sun Mar 31 13:50:45 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   58C    P8              10W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

Google Colab

- 라이브러리 확인
  - Python, tensorflow, matplotlib 등이 미리 설치되어 있음

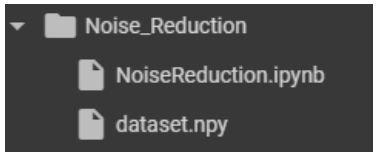
```
0초
1 !python --version
Python 3.10.12
```

```
2초
1 !pip list
markdown-it-py          3.0.0
MarkupSafe              2.1.5
matplotlib              3.7.1
matplotlib-inline       0.1.6
matplotlib-venn         0.11.10
mdit-py-plugins         0.4.0
mdurl                   0.1.2
```

```
2초
1 !pip list
tenacity                        8.2.3
tensorboard                     2.15.2
tensorboard-data-server         0.7.2
tensorflow                      2.15.0
tensorflow-datasets             4.9.4
tensorflow-estimator            2.15.0
tensorflow-gcs-config           2.15.0
tensorflow-hub                  0.16.1
tensorflow-io-gcs-filesystem    0.36.0
tensorflow-metadata             1.14.0
tensorflow-probability          0.23.0
tensorstore                     0.1.45
```

Google Colab

- 데이터셋 로드 방법 ①
  - 제공된 dataset.npy 파일 구글 드라이브에 저장
  - 구글 드라이브 연동 수행




```
[1]  1 from google.colab import drive
     2 drive.mount('/content/drive')

     Mounted at /content/drive
```

  - 다음 코드에 dataset.npy 저장 경로 수정하여 데이터 로드

```
 1 import math
 2 import numpy as np
 3 import tensorflow as tf
 4 from tensorflow import keras
 5 import matplotlib.pyplot as plt
 6
 7
 8 if __name__ == '__main__':
 9
10   dataset_resize  = np.load('/content/drive/MyDrive/Colab Notebooks/Noise_Reduction/dataset.npy')
11   train_data      = dataset_resize[:3000]
12   test_data       = dataset_resize[3000:]
13   data_shape      = train_data[0].shape
14
15   print(train_data.shape)
16   print(test_data.shape)
17
18   plt.imshow(train_data[0], cmap='gray')
19   plt.show()
20   plt.imshow(test_data[0], cmap='gray')
21   plt.show()
```

Google Colab

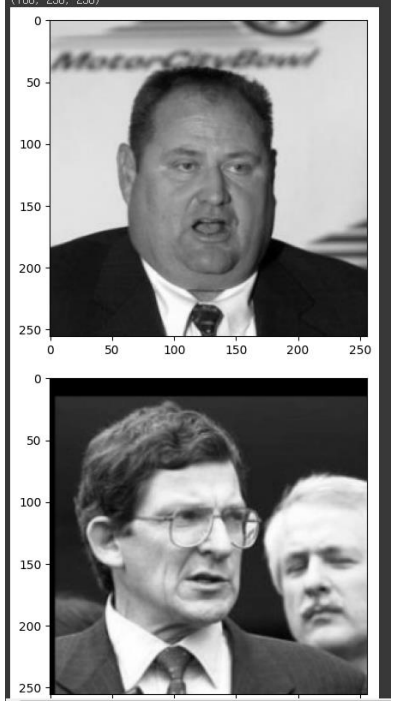
- 데이터셋 로드 방법 ②

  * 둘 중 한가지 방법을 통해 데이터셋 로드 수행

  – 다음 코드를 통해,
  Tensorflow-datasets로 lfw datasets 직접 다운로드 후 전처리 수행

```python
import math
import numpy as np
import tensorflow as tf
from tensorflow import keras
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
from skimage.transform import resize


if __name__ == '__main__':

  dataset, info   = tfds.load('lfw', split='train[:3100]', with_info=True)
  dataset_list    = [img['image'] for img in tfds.as_numpy(dataset)]
  dataset_color   = np.array(dataset_list)
  dataset_gray    = np.dot(dataset_color[..., :3], [0.299, 0.587, 0.114]).astype(np.uint8)
  dataset_resize  = (resize(dataset_gray, (3100, 256, 256))*255).astype(np.uint8)
  train_data      = dataset_resize[:3000]
  test_data       = dataset_resize[3000:]
  data_shape      = train_data[0].shape

  print(train_data.shape)
  print(test_data.shape)

  plt.imshow(train_data[0], cmap='gray')
  plt.show()
  plt.imshow(test_data[0], cmap='gray')
  plt.show()
```

```
Downloading and preparing dataset 172.20 MiB (download: 172.20 M
Dl Completed...: 100%  1/1 [01:55<00:00, 71.51s/ url]
Dl Size...: 100%  172/172 [01:55<00:00,  2.47 MiB/s]
Extraction completed...: 100%  13233/13233 [01:55<00:00, 708.93 fil
Dataset lfw downloaded and prepared to /root/tensorflow_datasets
(3000, 256, 256)
(100, 256, 256)
```

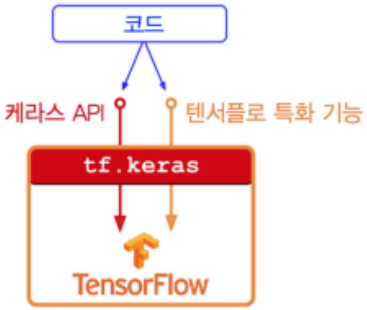


TENSORFLOW KERAS

Tensorflow Keras

- Tensorflow
  - 머신 러닝용 오픈소스 프레임워크
  - 데이터 플로우 그래프를 사용하여 수치 연산을 하는 오픈소스 소프트웨어 라이브러리
  - 구글에서 개발한 비공개 라이브러리였던 DistBelief에서 발전한 오픈소스 라이브러리

- Keras
  - Keras는 딥러닝 모델을 빌드하고 학습시키기 위한 Tensorflow의 high-level API



Tensorflow Keras

- Keras
  - 딥러닝 모델을 빌드하고 학습시키기 위한 Tensorflow의 상위 수준 API

```
import tensorflow as tf
from tensorflow import keras
```

- Sequential 모델
  - 각 레이어에 하나의 입력 텐서와 출력 텐서가 있는 일반 레이어 스택

```
# Define Sequential model with 3 layers
model = keras.Sequential(
    [
        keras.layers.Dense(units=4, activation=‘relu’, name="layer1"),
        keras.layers.Dense(units=8, activation=‘relu’, name="layer2"),
        keras.layers.Dense(units=8, activation='sigmond’, name="layer3"),
    ]
)
# Call model on a test input
x = tf.ones((10, 1))
y = model(x)
```

Tensorflow Keras

- Summary() 함수
  - 모델 요약

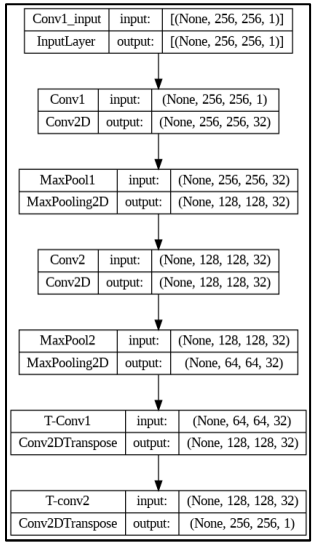
```
model.summary()
```

- plot_model() 함수
  - 모델의 각 레이어의 입력 및 출력 모양 표시

```
keras.utils.plot_model(model, "model_with_shape_info.png", show_shapes=True)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv1 (Conv2D)              (None, 256, 256, 32)      320

 MaxPool1 (MaxPooling2D)     (None, 128, 128, 32)      0

 Conv2 (Conv2D)              (None, 128, 128, 32)      9248

 MaxPool2 (MaxPooling2D)     (None, 64, 64, 32)        0

 T-Conv1 (Conv2DTranspose)   (None, 128, 128, 32)      9248

 T-conv2 (Conv2DTranspose)   (None, 256, 256, 1)       289

=================================================================
Total params: 19,105
Trainable params: 19,105
Non-trainable params: 0
_________________________________________________________________
```

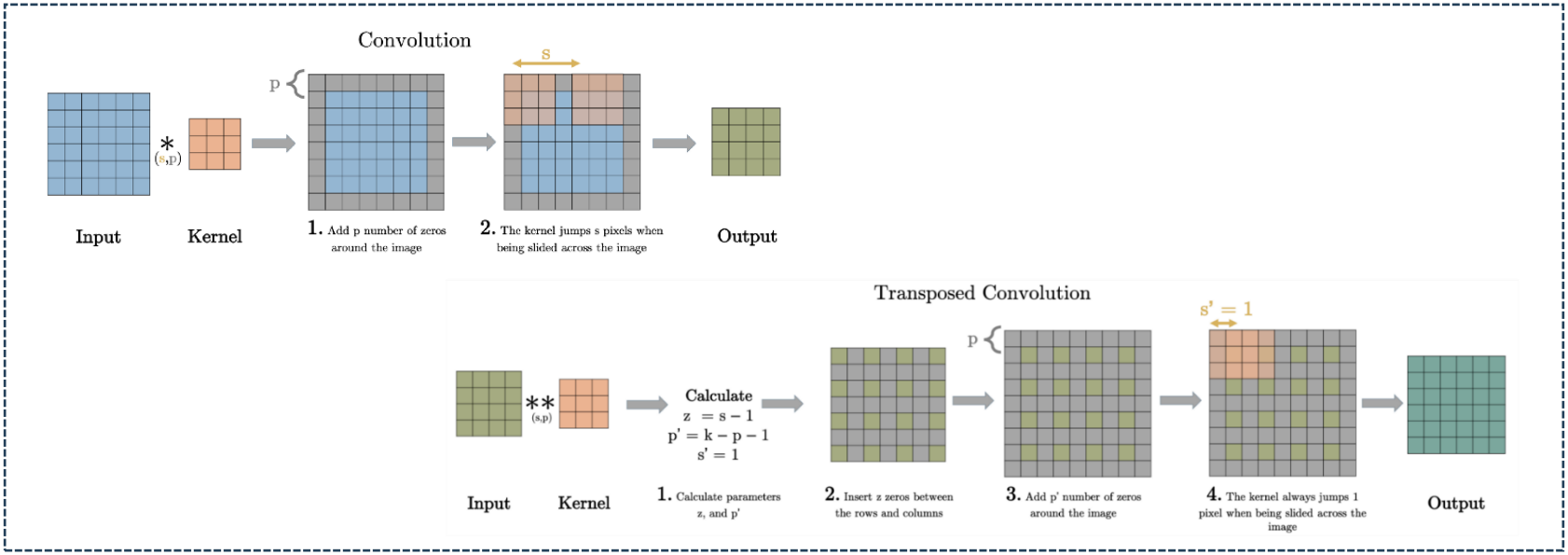
summary 함수 예시

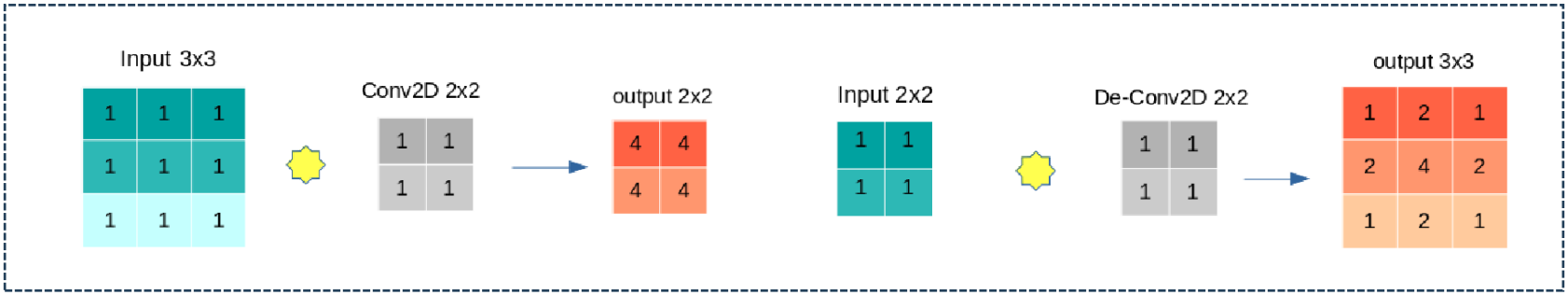


plot_model 함수 예시

Tensorflow Keras

- 2D Convolution layer & 2D Transposed Convolution layer 예시





Tensorflow Keras

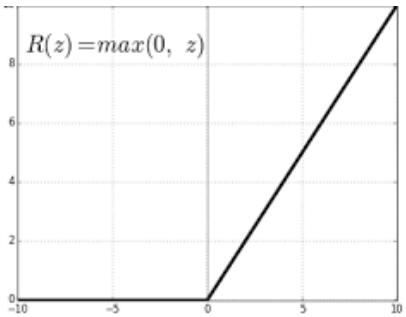
- Conv2D & Conv2DTranspose

```
keras.layers.Conv2D(filters,  kernel_size, strides=(1, 1), padding="valid", activation=None, input_shape, name)
```

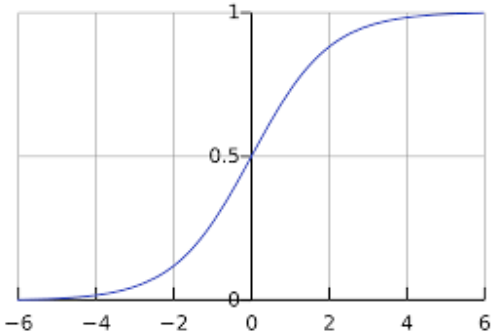
```
tf.keras.layers.Conv2DTranspose(filters,  kernel_size, strides=(1, 1), padding="valid", activation=None, input_shape, name)
```

– 주요 파라미터
  - filters: 컨볼루션 필터의 수,
  - strides: 커널의 이동 간격 설정,
  - activation: 활성화 함수 설정,

kernel_size: 컨볼루션 커널의 (행,열)
padding: 패딩 옵션
input_shape: 입력 형태



ReLU



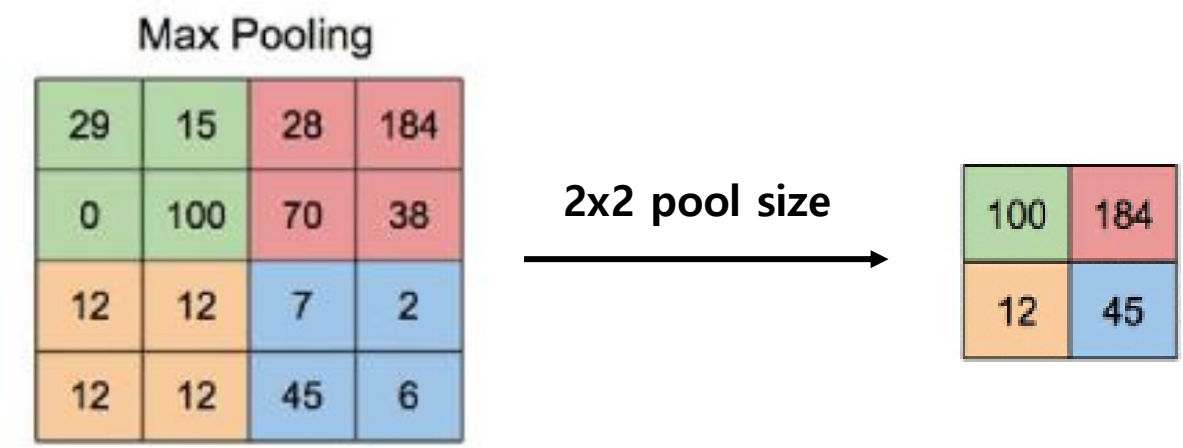
Sigmoid

활성화 함수 예시

Tensorflow Keras

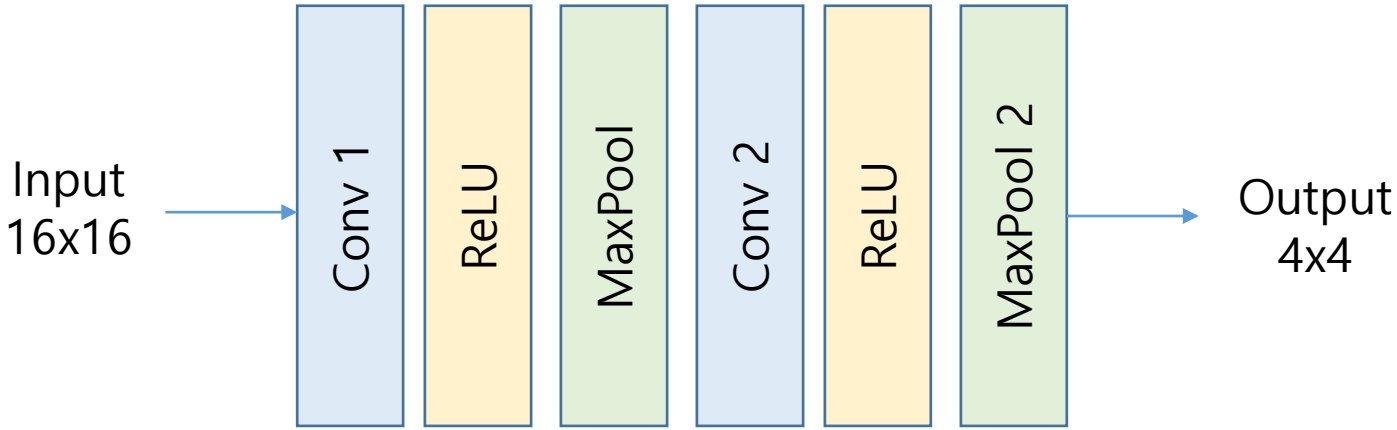
- Maxpooling2D
  - 공간적 데이터에 대한 최대값 풀링 작업
  - 파라미터
    - pool_size: 윈도우 patch의 수직, 수평 축소 비율 지정

```
tf.keras.layers.MaxPooling2D( pool_size=(2, 2), strides=None, padding="valid", name)
```



Tensorflow Keras

- CNN architecture 실행 예제
  - Conv2D
    - 필터 수: 16 / Kernel size: (3x3) / Activation: Relu
  - Maxpooling2D
    - pool_size: (2x2)




```
x = keras.layers.Conv2D(16, (3, 3), activation="relu", padding="same")(input)
x = keras.layers.MaxPooling2D((2, 2), padding="same")(x)
x = keras.layers.Conv2D(16, (3, 3), activation="relu", padding="same")(x)
x = keras.layers.MaxPooling2D((2, 2), padding="same")(x)
```

Tensorflow Keras

- 교육, 평가 및 추론
  - compile() 함수
    - 손실 함수, 최적화 방법 등을 선택적으로 지정

```
model.compile( loss, optimizer )
```

  - 훈련 fit() 함수
    - 데이터를 "batch_size" 크기로 분할하고 지정된 수의 "epoch"에 대해 전체 데이터세트를 반복 처리하여 모델을 훈련
      - x: 입력 데이터, y: 타겟 데이터

```
model.fit( x, y, batch_size, epochs )
```

  - 추론 evaluate() 함수
    - 테스트 데이터에 대해 모델 평가
      - x: 입력 데이터, y: 타겟 데이터
      - 데스트 데이터에 대한 손실함수 값 반환

```
model.evaulate(x_test, y_test, verbose=2)
```

Tensorflow Keras

- 데이터셋 normalize
  - 딥러닝 학습 간 normalize 하여 입력
    - 데이터값 범위 : 0 ~ 255 → 0 ~ 1
    - Keras layer 입력 형태 : (batch_size, width, height, channel)

```
# Normalize data (to get 0 ~ 1 range)
train_data        = (train_data.astype(np.float32) / 255.).reshape(3000, 256, 256, 1)
train_data_noise  = (train_data_noise.astype(np.float32) / 255.).reshape(3000, 256, 256, 1)
test_data         = (test_data.astype(np.float32) / 255.).reshape(100, 256, 256, 1)
test_data_noise   = (test_data_noise.astype(np.float32) / 255.).reshape(100, 256, 256, 1)
```

END OF PRESENTATION

Q&A