연구 노트

04.01

- 수행 계획서 제출
- 주제: 소행성 3D 형상 모델링
- 선행 연구 조사:
  - NASA- Deep Asteroid Software Model
  - THOR(Tracklet-less Heliocentric Orbit Recovery)
  - ATLAS
  - KOALA(Knitted Occultation, Adaptive-optics, and Lightcurve Analysis)
  - ADAM

04.29

- 문제점: 소행성 3D 형상 모델의 경우 광도곡선 역산법의 등장 이래로 유의미한 발전을 이룩하지 못하였으며, THOR의 경우 30일간의 지속적인 관측데이터가 요구되는 등 소행성에 대한 세부적인 분석에 있어서 한계를 가진다.
  $\rightarrow$ LSTM, GAN 등의 적절한 융합으로 광도곡선 to 소행성 모델링이 가능하지 않을까?
- 조사
  - 1. GAN
  - 두 네트워크 G, D가 번갈아가며 경쟁적으로 **Unsupervised** learning:
  - Generator: Random noise로부터 데이터 생성
  - Discriminator: G 출력의 진위 여부 판별
  - G: 정규분포에서 추출한 무작위 텐서로부터 실제 데이터의 복잡한 분포 학습.

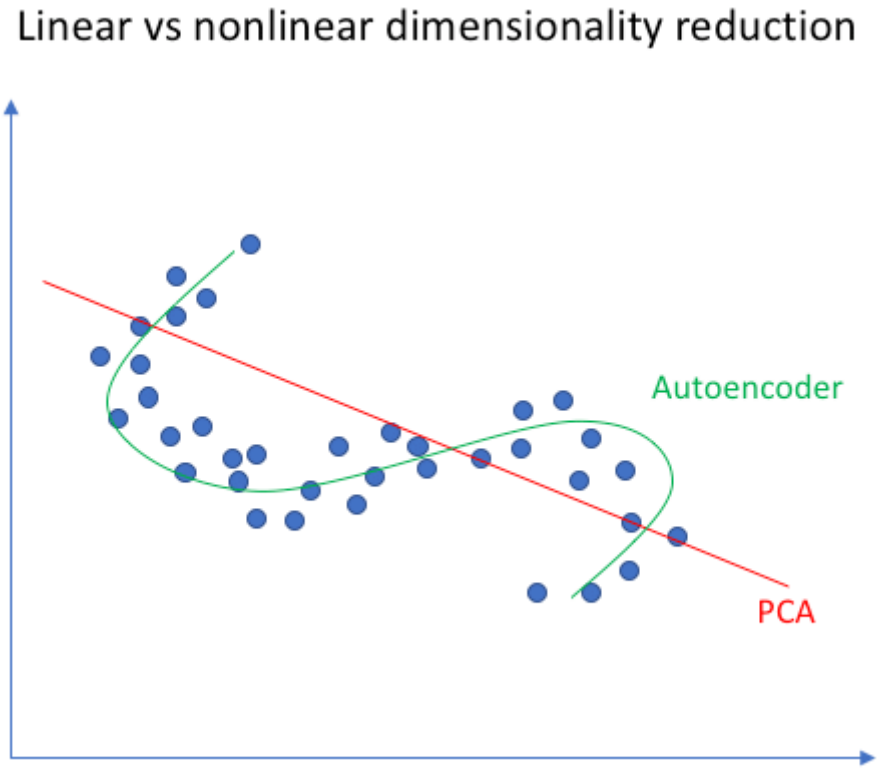
    ```
    - 학습이 불안정함. 대규모 데이터셋으로 학습시키기 힘듦.
    - 비지도학습의 한 종류지만 cGAN은 지도학습에 활용될 수 있다.
    ```

이미지는 고차원(x, y, channel) latent space에서의 한 점으로 표현된다. 즉 생성하고자 하는 이미지의feature는 latent space에서 다변수 확률분포를 나타낼 수 있는데, GAN(Generative Adversial Network)은 이를 이용하여 Generator를 출력하고자 하는 이미지의 확률분포에 수렴시키는 것을 목표로 한다. GAN은 Generator(G) 신경망과 Discriminator(D) 신경망을 경쟁적으로 번갈아가며 학습시키므로써 데이터셋으로부터 원하는 자료를 생성할 수 있도록 하며 목적함수는 아래와 같다.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

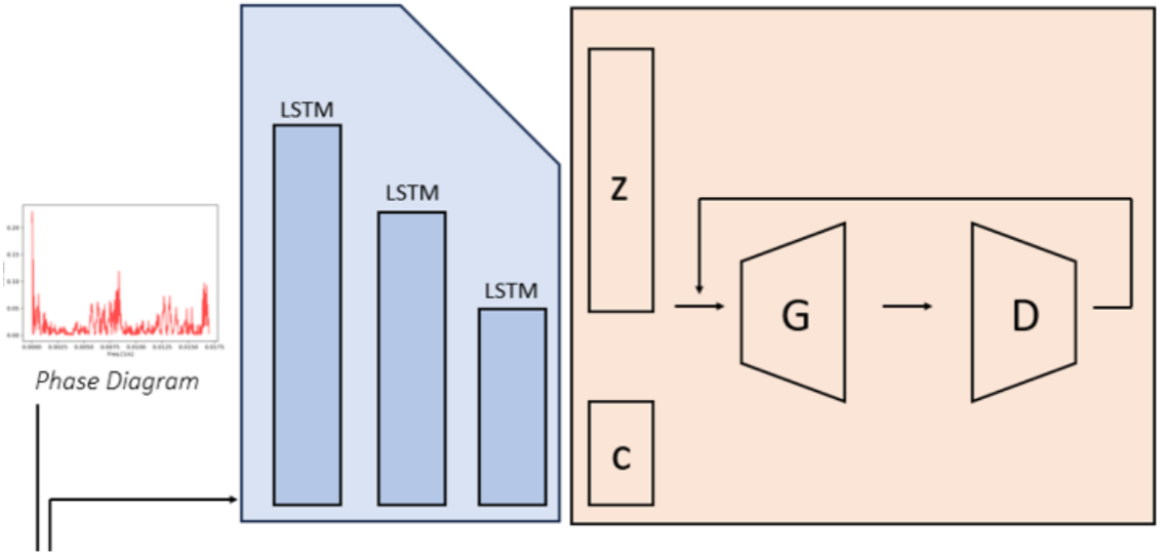
D는 데이터셋의 확률분포 $p_{data}(x)$로부터 x를 샘플링하는 사건에 대한 로그값의 평균을 통해 얼마나 데이터셋과 유사한지 0과 1 사이 값으로 평가하고, G는 Noise $p_z(z)$로부터 D를 통해 실제 데이터의 분포에 근사시킴을 알 수 있다. GAN은 데이터셋의 분포만 사용하기에 비지도학습으로 분류되며 학습 이후 G만 사용한다. 한편 cGAN은 정규분포에서 샘플링한 z 벡터에 conditioning 벡터를 추가하여 GAN의 학습방향을 지도한다.

- AE
  - Reconstruction Loss: AE의 Loss
  - 신경망 차원을 줄이다가 키움으로써 Bottleneck을 유도하여 feature 압축.
  → latent vector/variable/feature/code, hidden representation
  - x가 Input이자 Label이다.
  - PCA와 달리 비선형적으로 차원을 축소시킴.



05.23

- AI 모델 구상



[LSTM-AE와 3D-cGAN을 연결한 전체 아키텍처]

  3D-VAE-GAN은 VAE의 Decoder와 GAN의 Generator가 연결되어 있는 구조로, Encoder E, Decoder이자 Generator G, Discriminator D로 구성된다.
- 2D Image와 그에 대응되는 3D 모델로 **supervised** learning, E와 G 학습.
- Mesh: 채색을 위한 grid. 더 복잡하고 유연한 형상을 가짐.

05.28

- KASI 컨설팅
- 용어 정리
- Filter: Operation toward tensor data types: img, vid, ...
- Noise는 고주파 성분으로, LPF를 통해 걸러낸다.
- HPF는 고주파 성분에 해당하는 edge를 강화하고 저주파 성분에 해당하는 background를 지운다.
  - e.g. Laplacian filter: 이계도함수를 이용하기에 noise에 민감해서 충분히 blurring한 후 사용해야.
- Edge detection은 gradient만으로도 꽤 잘 됨.
- Bandpass filter: 특정 구간의 주파수 성분만 통과시킴.
- Notch filter: 특정 구간의 주파수 성분만 차단시킴.
- Gradient Removal/Elimination: unwanted gradient나 background variation을 제거.

06.07

(126) Velleda, (130) Elektra 원격 관측 제안서 DOAO NYSC에 제출.

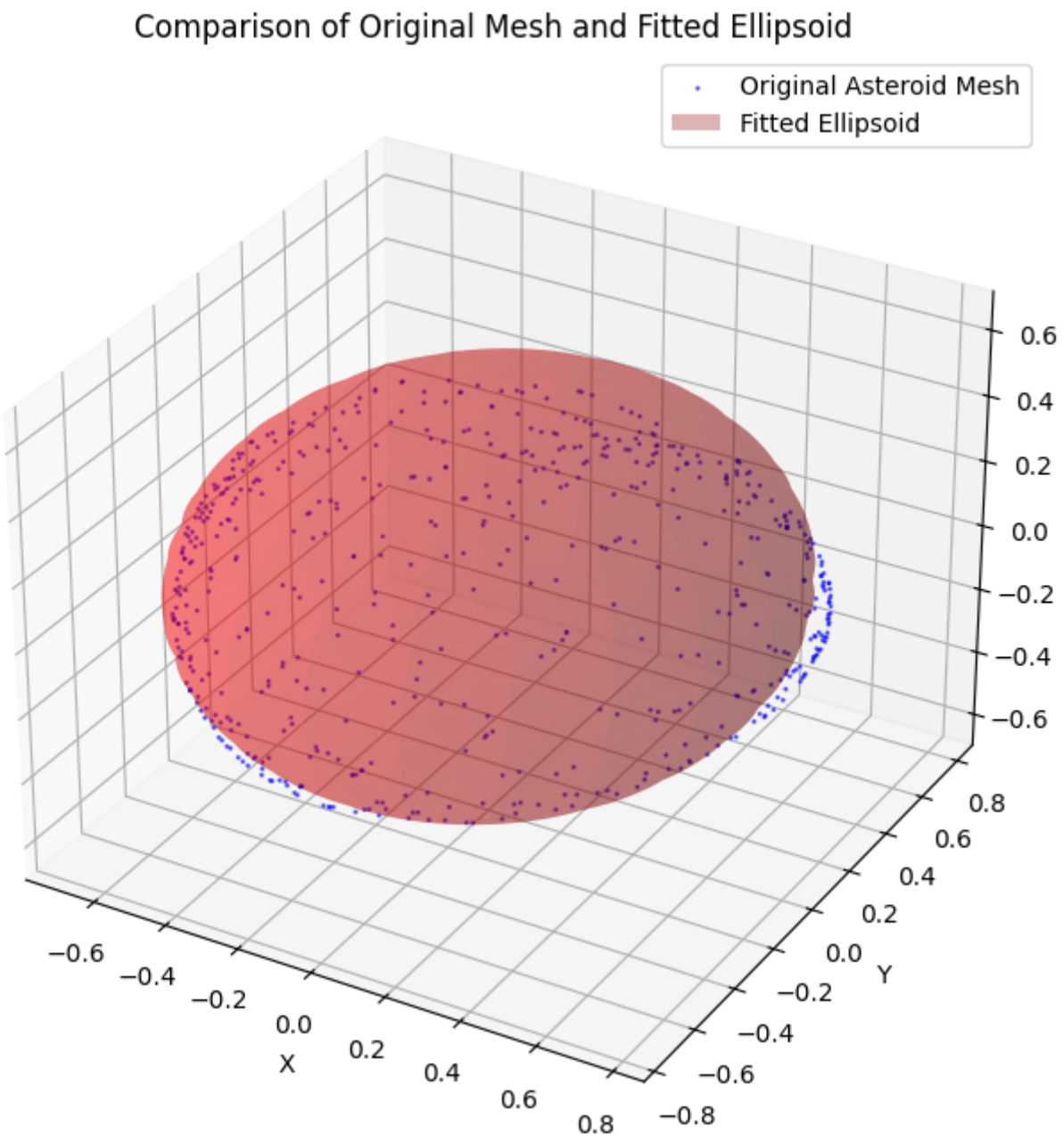
07.07

Ellipsoid fitting 코드 작성

```python
import os

import numpy as np

import trimesh

from scipy.optimize import least_squares

from scipy.spatial.transform import Rotation as R

from tqdm import tqdm

import concurrent.futures

import multiprocessing

# 1. Ellipsoid 피팅 함수 정의

def ellipsoid_model(params, x, y, z):

a, b, c = params[0:3]

cx, cy, cz = params[3:6]

alpha, beta, gamma = params[6:9]

R_mat = R.from_euler('xyz', [alpha, beta, gamma]).as_matrix()

xyz = np.vstack((x - cx, y - cy, z - cz))

rotated_xyz = R_mat @ xyz

X, Y, Z = rotated_xyz

return (X / a)**2 + (Y / b)**2 + (Z / c)**2 - 1

# Ellipsoid 피팅 함수 정의

def fit_ellipsoid(points):

x, y, z = points[:, 0], points[:, 1], points[:, 2]

a0, b0, c0 = np.std(x), np.std(y), np.std(z)

cx0, cy0, cz0 = np.mean(x), np.mean(y), np.mean(z)

angles0 = [0, 0, 0]

initial_params = [a0, b0, c0, cx0, cy0, cz0] + angles0
```

```
# 매개변수 경계 설정

lower_bounds = [1e-3, 1e-3, 1e-3, -np.inf, -np.inf, -np.inf, -np.pi, -np.pi,
-np.pi]

upper_bounds = [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.pi,
np.pi, np.pi]

result = least_squares(ellipsoid_model, initial_params, args=(x, y, z),
bounds=(lower_bounds, upper_bounds))

return result.x

# 파일 처리 함수 정의 (Ellipsoid만 변환)

def process_file(obj_file):

try:

# 각 출력 파일의 경로 설정

ellipsoid_path = os.path.join(output_dirs['ellipsoid'],
obj_file.replace('.obj', '_ellipsoid.npy'))

file_path = os.path.join(input_directory, obj_file)

mesh = trimesh.load(file_path)

points = mesh.vertices

# Ellipsoid 변환 및 저장

ellipsoid_params = fit_ellipsoid(points)

np.save(ellipsoid_path, ellipsoid_params)

except Exception as e:

print(f"Error processing {obj_file}: {e}")

if __name__ == "__main__":

# 경로 설정

input_directory = '/content/drive/MyDrive/output_data_2' # Colab 환경에 맞게
수정하세요

output_dirs = {
```

```
'ellipsoid': '/content/drive/MyDrive/output_ellipsoid'

}

# 결과를 저장할 폴더 생성

for key, path in output_dirs.items():

if not os.path.exists(path):

os.makedirs(path)

# obj 파일 리스트 가져오기

obj_files = [f for f in os.listdir(input_directory) if f.endswith('.obj')]

# 병렬 처리 실행

num_workers = multiprocessing.cpu_count()

# Colab 환경에서는 ThreadPoolExecutor를 사용하는 것이 더 안정적일 수 있습니다.

with concurrent.futures.ThreadPoolExecutor(max_workers=num_workers) as
executor:

list(tqdm(executor.map(process_file, obj_files), total=len(obj_files),
desc="Processing .obj files"))
```



08.06

- Elektra 원격 관측 수행.
- Dark, Bias, Flat 추가 관측.

08.20 KASI 2차 방문

아예 png로 이미지로 그냥 가버리는 것보다, lightcurve inversion 자체의 inductive bias를 활용하는 건 어떨까?
MLP를 사용할 수 있는데, 최근에 발표된 KAN을 사용하면 feature attribution score 등 interpretable하다는 특징을 적극적으로 활용하여 더 깊은 분석이 가능해보인다.

09.25~10.05

- 타원체 근사에 대한 lightcurve inversion 유도

소행성에서 Lambertian 반사 및 균일한 알베도를 가정했을 때, 관측된 밝기 $ B *와사영된영역*$A(\theta, \phi)$가 비례한다고 알려져 있다. 한편 Cauchy's surface area formula에 의하면, $A(\theta, \phi)$는 아래와 같이 표기할 수 있다. (θ, ϕ)는 각각 z축과 시선 벡터 간 각도, x축과 시선 벡터 간 각도이다.

$$A(\theta, \phi) = \int_0^{\pi} \int_0^{2\pi} \mathbf{n} \cdot \mathbf{v} \, dS$$

여기서 unit normal vector $\mathbf{n}$과 시선 벡터 $\mathbf{v}$는 $F = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1$에 대해 아래와 같이 정리된다.

$$\mathbf{n} = \frac{\nabla F}{\|\nabla F\|} = \frac{\left(\frac{2x}{a^2}, \frac{2y}{b^2}, \frac{2z}{c^2} \right)}{\sqrt{\left(\frac{2x}{a^2} \right)^2 + \left(\frac{2y}{b^2} \right)^2 + \left(\frac{2z}{c^2} \right)^2}}, \quad \mathbf{v} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$$

그리고 $x = a\sin\theta'\cos\phi'$, $y = b\sin\theta'\sin\phi'$, $z = c\cos\theta'$로 매개화된 영역 $P(\theta', \phi')$에 대해,

$$dS = \left| \frac{\partial P}{\partial \theta'} \times \frac{\partial P}{\partial \phi'} \right| d\theta' d\phi'$$

라 나타내면 아래와 같다.

$$dS = \sin\theta' \sqrt{c^2 \sin^2\theta' (b^2\cos^2\phi' + a^2\sin^2\phi') + a^2 b^2 \cos^2\theta'} \, d\theta' d\phi'$$

이를 모두 고려하여 적분한 결과를 통해, 사영된 영역 및 밝기는 아래와 같이 구할 수 있다.

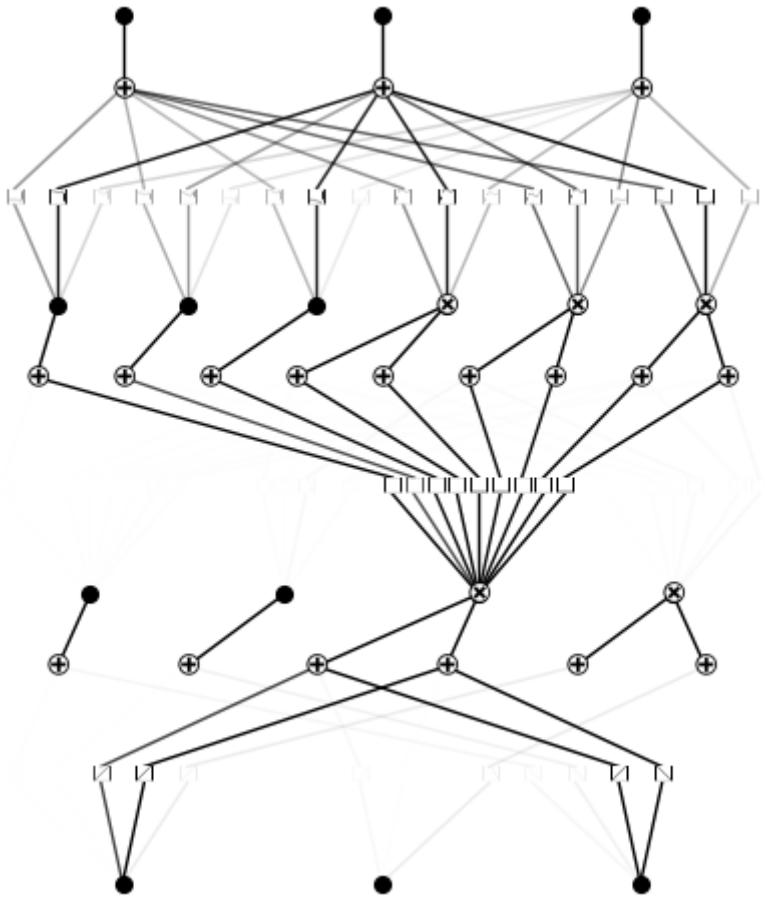
$$A(\theta, \phi) = \frac{\pi abc}{\sqrt{a^2\sin^2\theta\cos^2\phi + b^2\sin^2\theta\sin^2\phi + c^2\cos^2\theta}} \propto B(\theta, \phi)$$

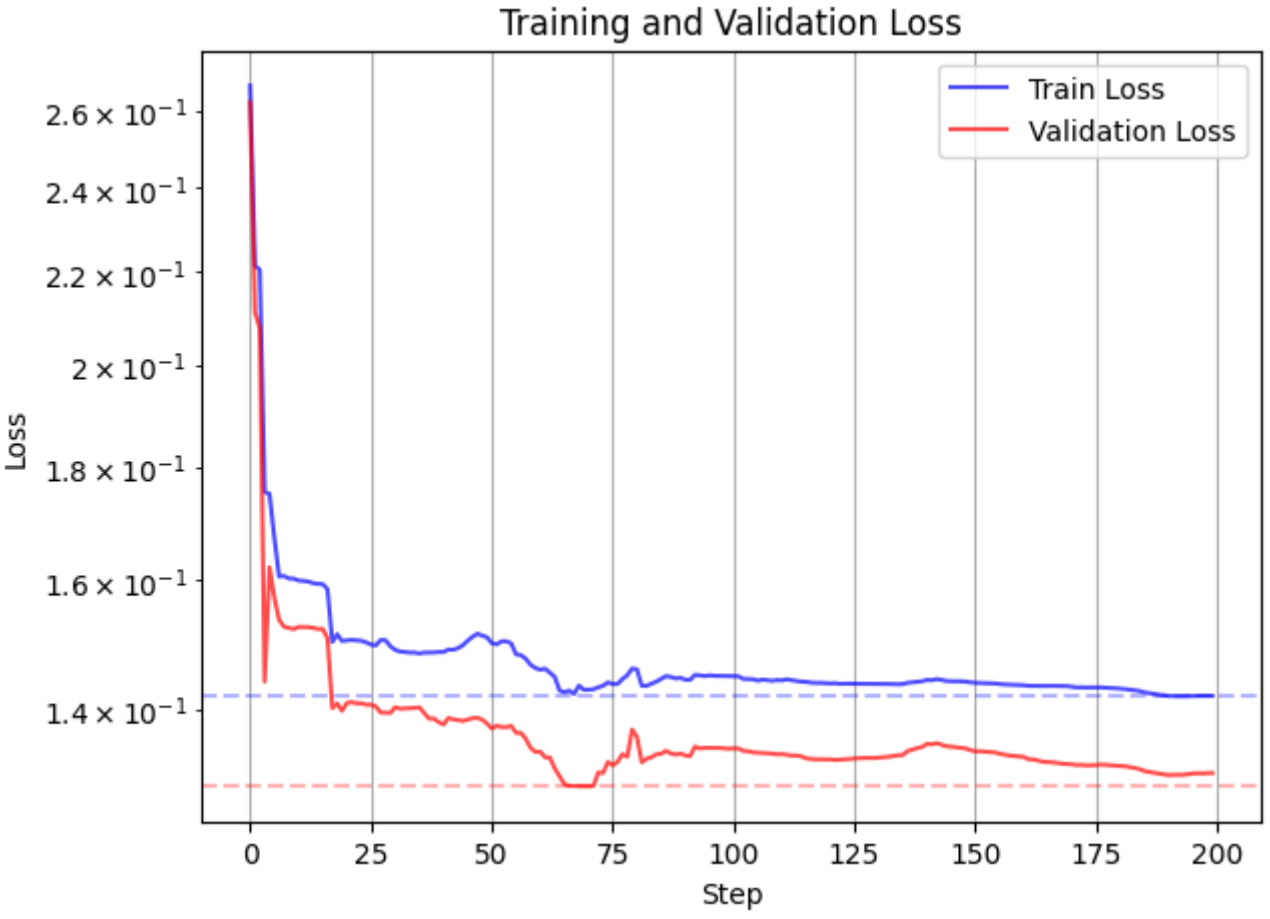
10.10

DAMIT dataset web crawling 코드 작성: 1280여개 특이해 하나만 존재하는 소행성 물리량 데이터셋 구성 완료.

11.07

- 최종 컨설팅
- KAN 훈련 완료 및 하이퍼파라미터 튜닝 완료료




Training and Validation Loss
Train Loss
Validation Loss
Loss
2.6×10^{-1}
2.4×10^{-1}
2.2×10^{-1}
2×10^{-1}
1.8×10^{-1}
1.6×10^{-1}
1.4×10^{-1}
0
25
50
75
100
125
150
175
200
Step

11.08

- KAN interpretation 및 주요 결과 정리