Hack presentations

Friday Sep 6 10-12

Each group 10 minutes presentation + 5 min Q&A

You may decide as a group how you structure your presentation (how many speakers, etc)

Please clearly lay out the motivation for your problem. Why is your problem interesting?

Please keep the A-H slides as demarcation markers.

A

AstroAI Asian Network (A3Net) Summer School

Sep 6, 2024

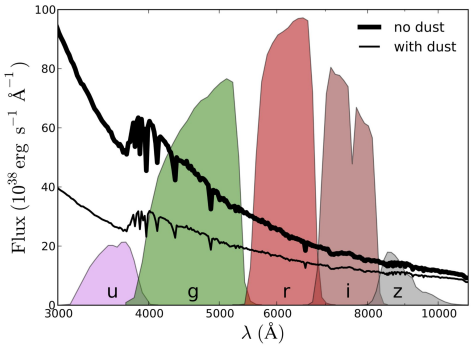
Hack 9: A generative model for spectra

Group A : Akhil Uniyal, Jongin Park, Lei haipeng, Takuma Moriyama, Yiming Huang, and Ziming Wang


01. What is our Mission?

Generating the Synthetic Galaxy Spectra

- Testing data processing and analysis pipeline for spectra
- The amount of observation spectra is limited
  → Realistic simulations of the anticipated data can be used
  → Perform learning with any generative model! (GAN, VAE, …)
- Our Goal is generating **synthetic galaxy spectra**, by learning from measured **Sloan Digital Sky Survey** data

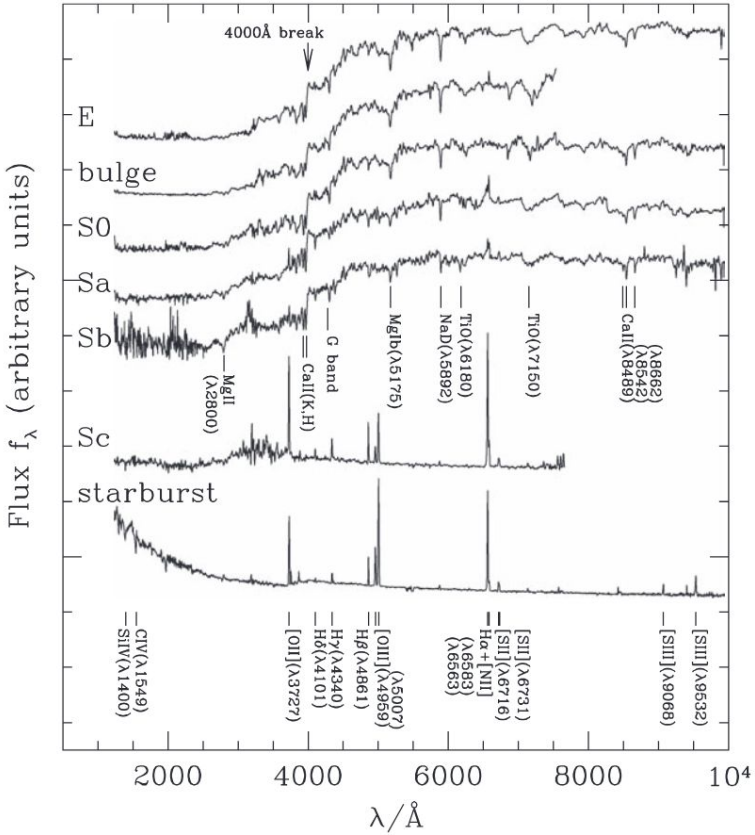


https://tao.asvo.org.au/tao/static/documentation/Technical-Spectral-Energy-Distribution-Module.html

02. Backgrounds

The Noticeable Feature of Galaxy Spectra

- **4000 Angstrom break:**
  - Sudden discontinuity in the spectra around 4000Å
  - This is due to sudden change in the opacity of low mass stars
  - This feature is predominantly in galaxies with stellar population ages greater than 0.1Gyr.

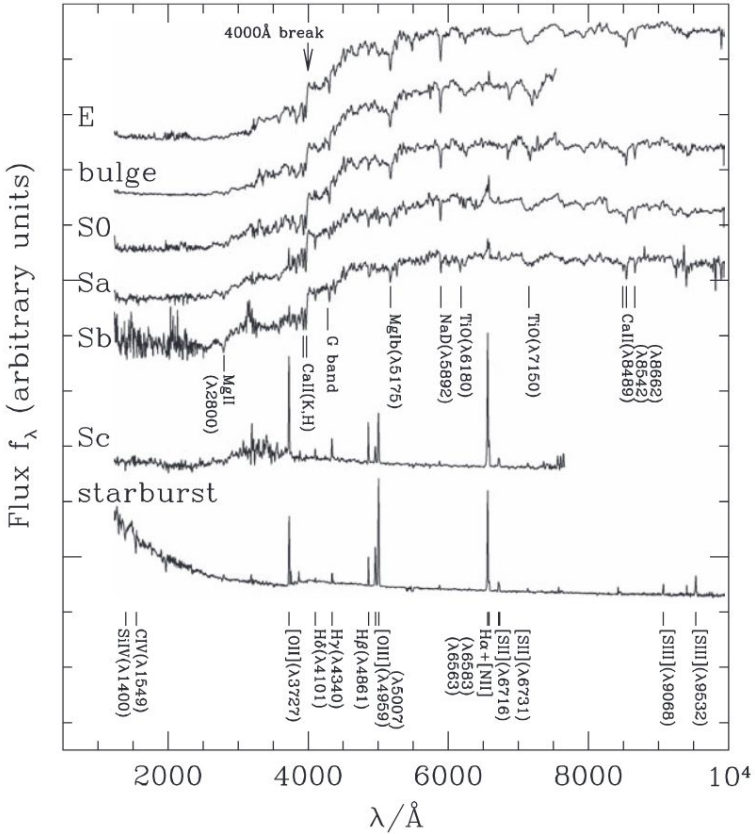


From Hojun Mo et al. “Galaxy Formation and Evolution”

The Noticeable Feature of Galaxy Spectra

- **Emission Line:**
  - Late type, starburst galaxies produce light in the blue and near ultraviolet from young stars
  - ISM get **ionized given rise to emission line**

- **Absorption line:**
  - Early type galaxies are lack of young stars, most of light emerges at the longest wavelengths.
  - Cold stars **giving rise to absorption lines.**

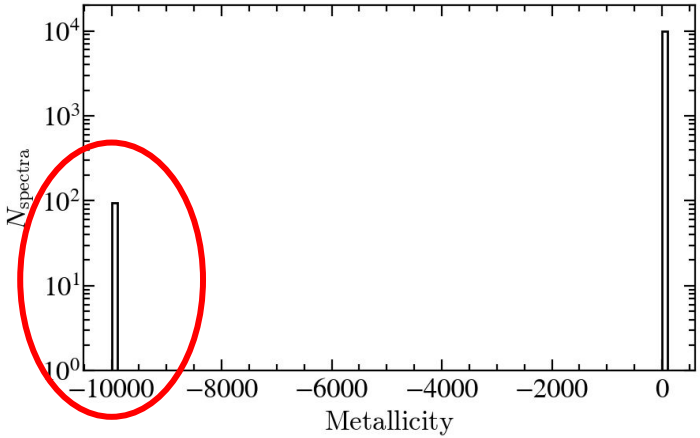


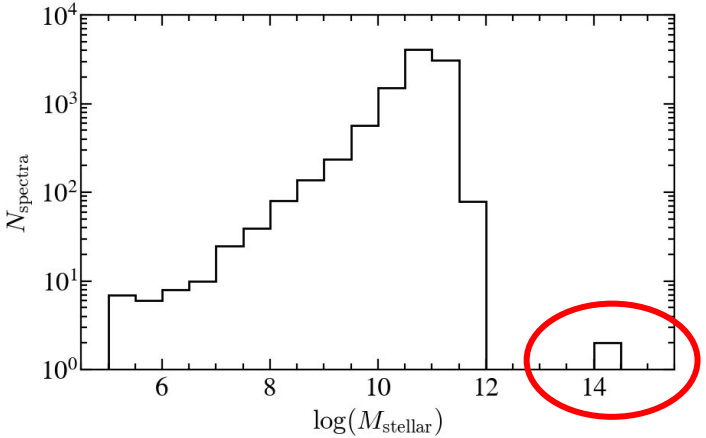
From Hojun Mo et al. “Galaxy Formation and Evolution”

03. Data Preprocess

Noticeable Issues on Datasets

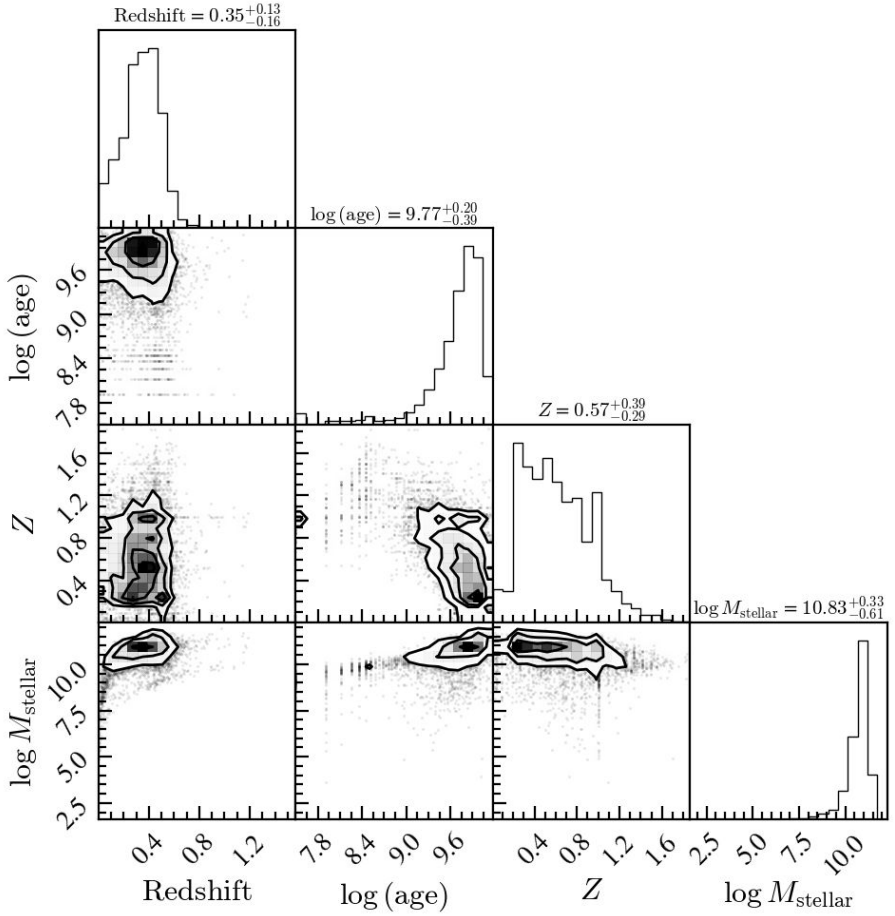
- Some spectra showed strange value in given physical parameters **(age, metallicity, smass)**
  some spectra showed…
  - negative redshift (5 spectras)
  - negative age, metallicity, smass (95 spectras)
  - smass massive than $10^{13} M_{\odot}$ (5 spectras)





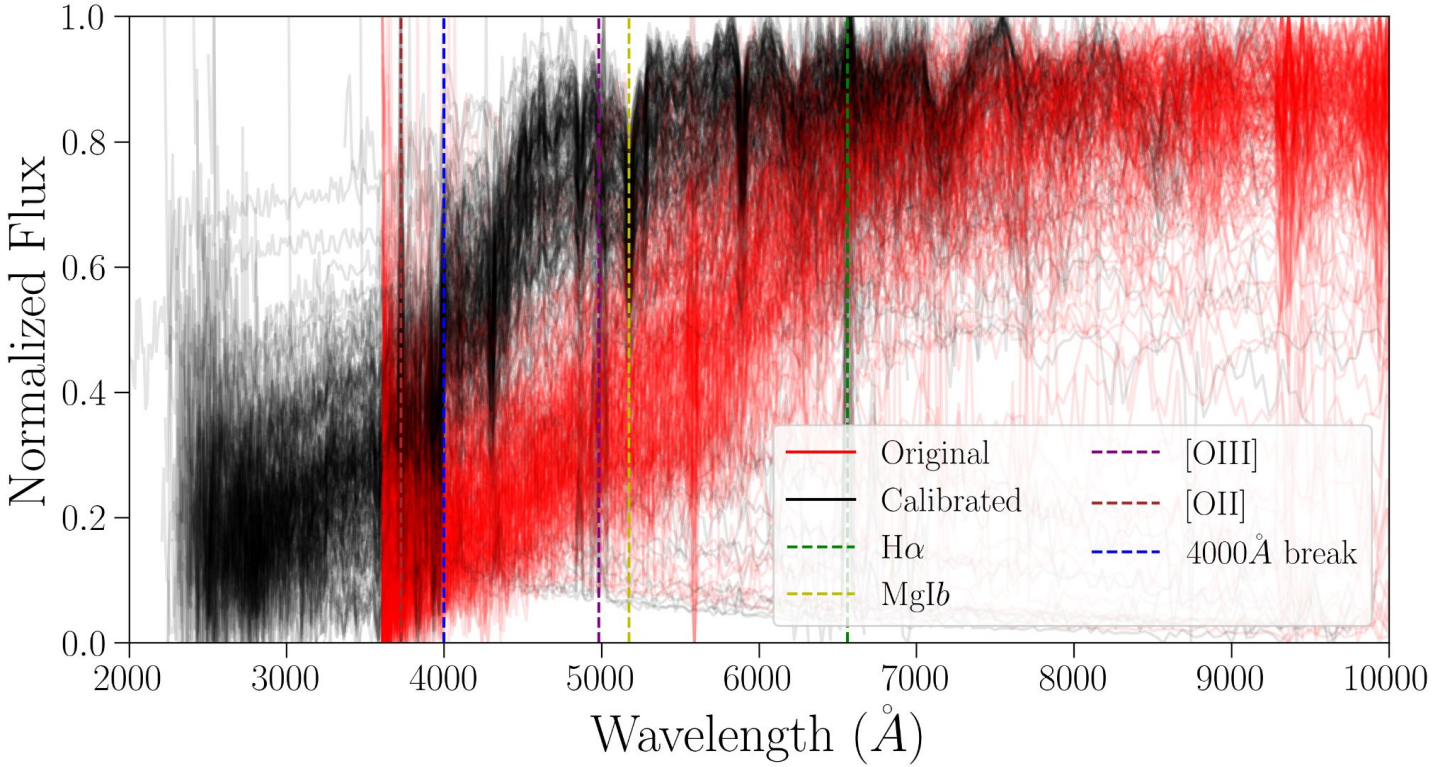
⇒ We removed unreasonable measurements from our datasets

Distribution of the Parameters after Removal

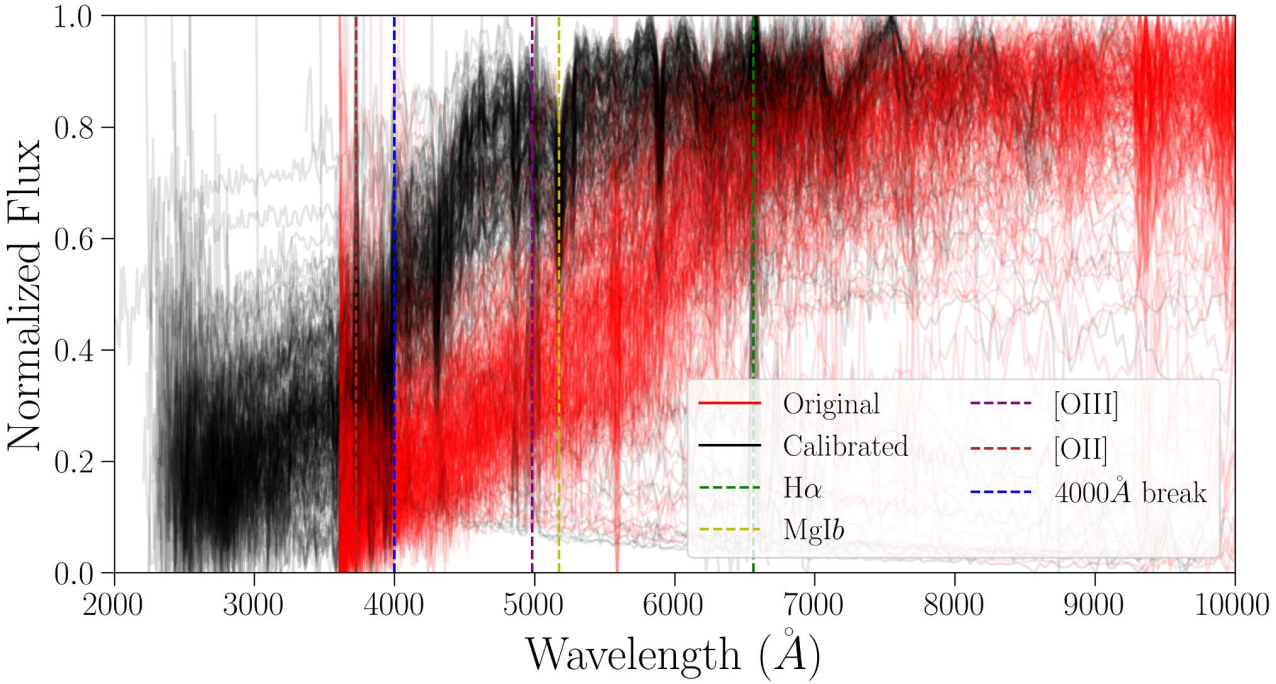


Spectra Wavelength Calibration

- The original spectra **(observer frame)**
  → Shift to **rest frame** based on redshift measurements



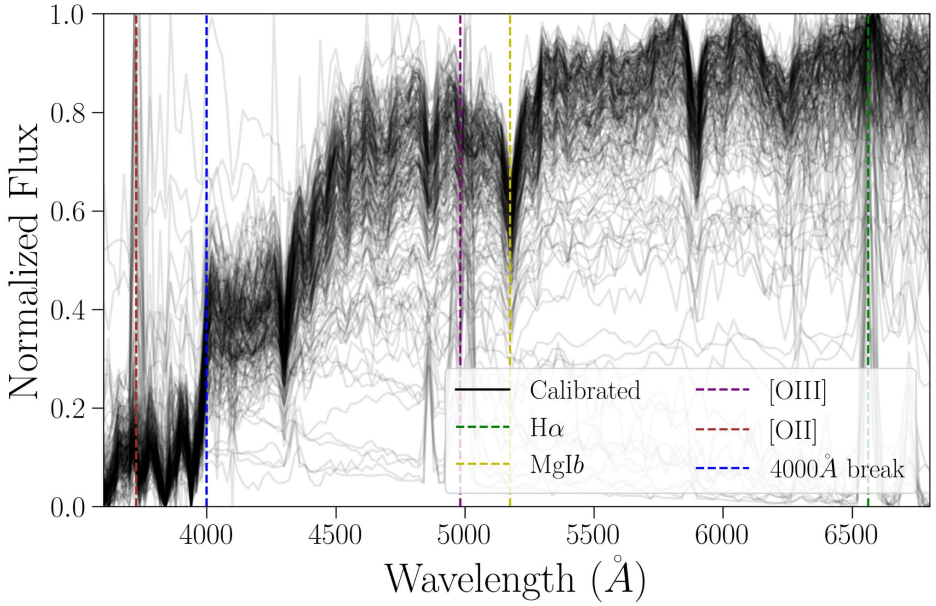
After Spectra Wavelength Calibration



- **Goods:**
  - Smaller scatter
  - Similar emission/absorption lines
- **Bads:**
  - Different wavelength range
  - Misaligned wavelength sampling points

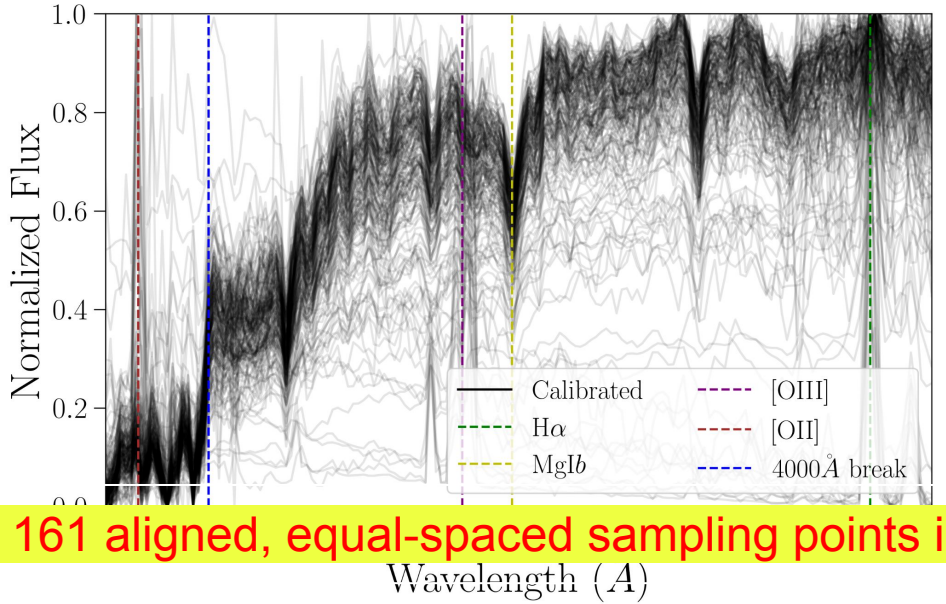
Alignment of Sampling Points in Spectra

- Different wavelength range problem
  → Extract the **overlapping parts** in different spectra
- Misaligned wavelength sampling points problem
  → Resample by **interpolating** within the overlapping range



Alignment of Sampling Points in Spectra

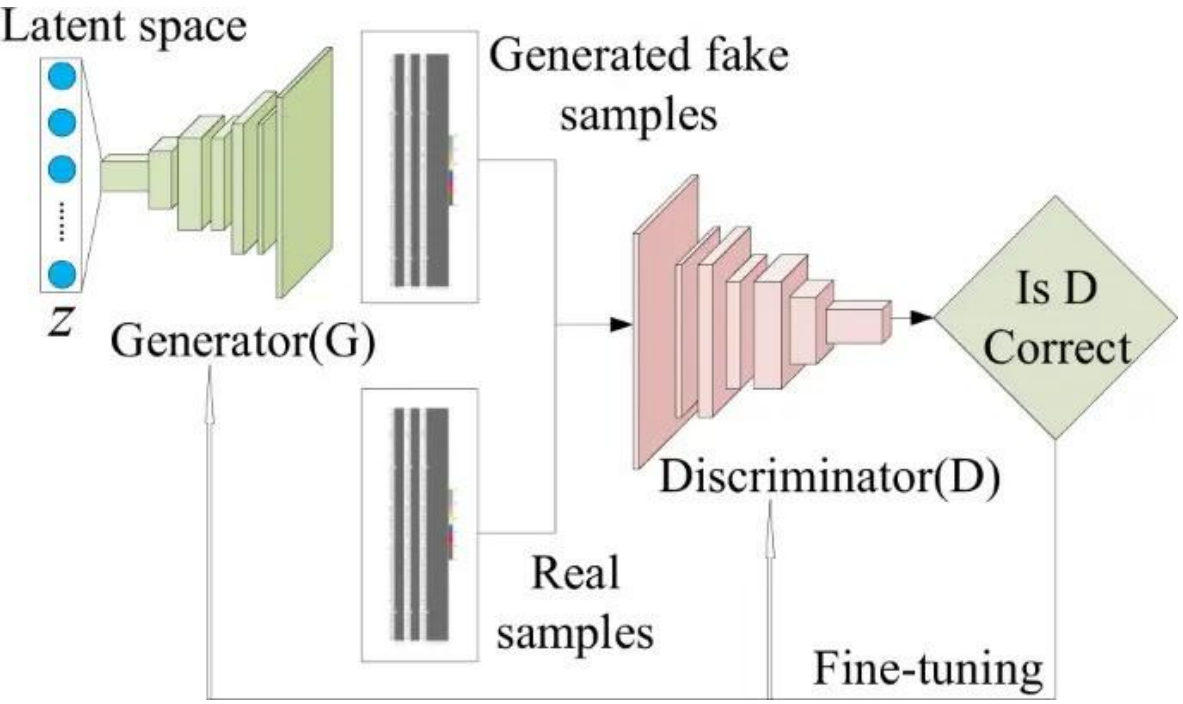
- Different wavelength range problem
  → Extract the **overlapping parts** in different spectra
- Misaligned wavelength sampling points problem
  → Resample by **interpolating** within the overlapping range



8824 spectra with 161 aligned, equal-spaced sampling points in 3600~6800 A

04. Semi-unsupervised GAN

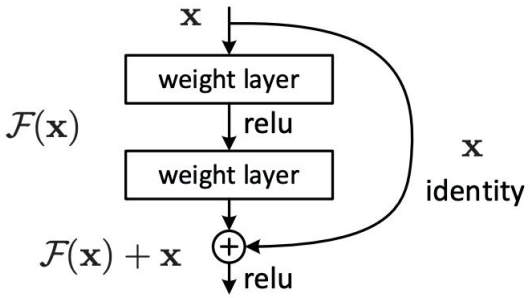
Generative Adversarial Networks (GAN)



https://www.labellerr.com/blog/what-is-gan-how-does-it-work/

Fighting , finally balance

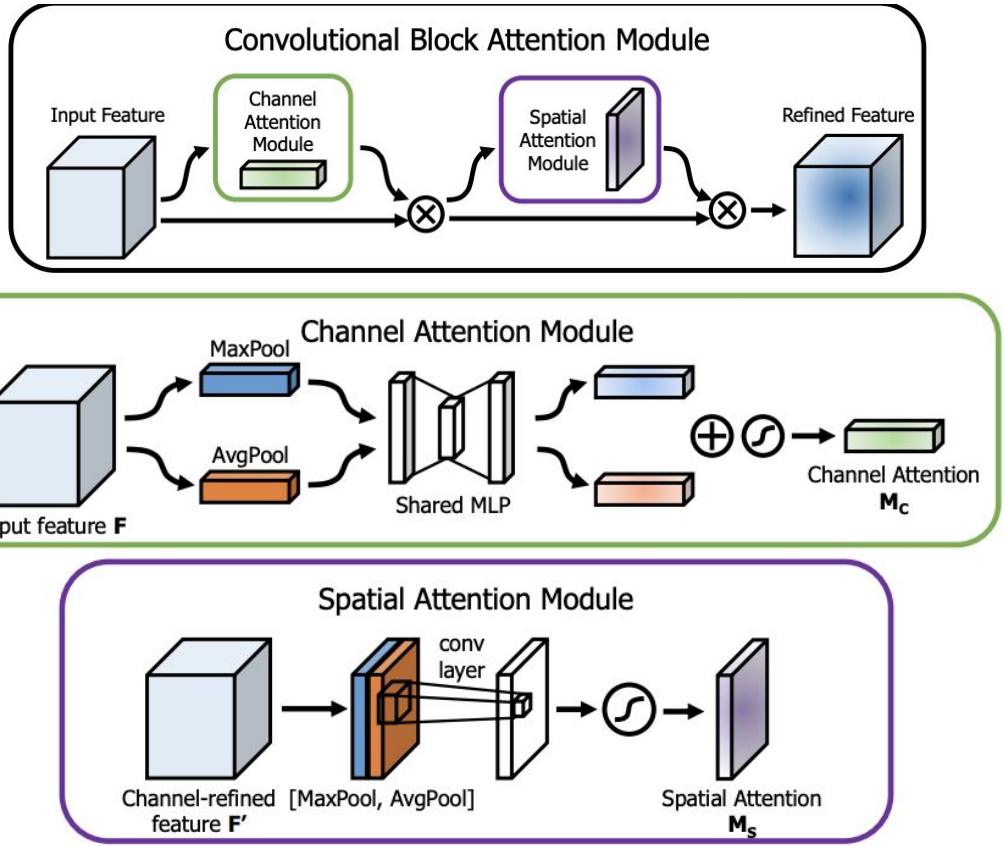
ResNet CBAM



Residual block

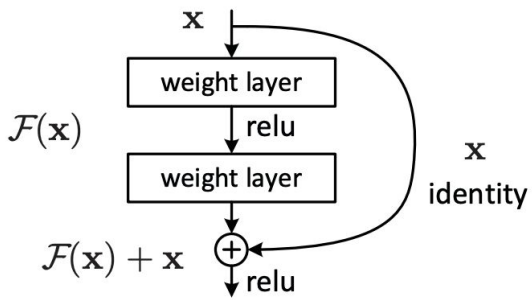
Kaiming He , et al 2015

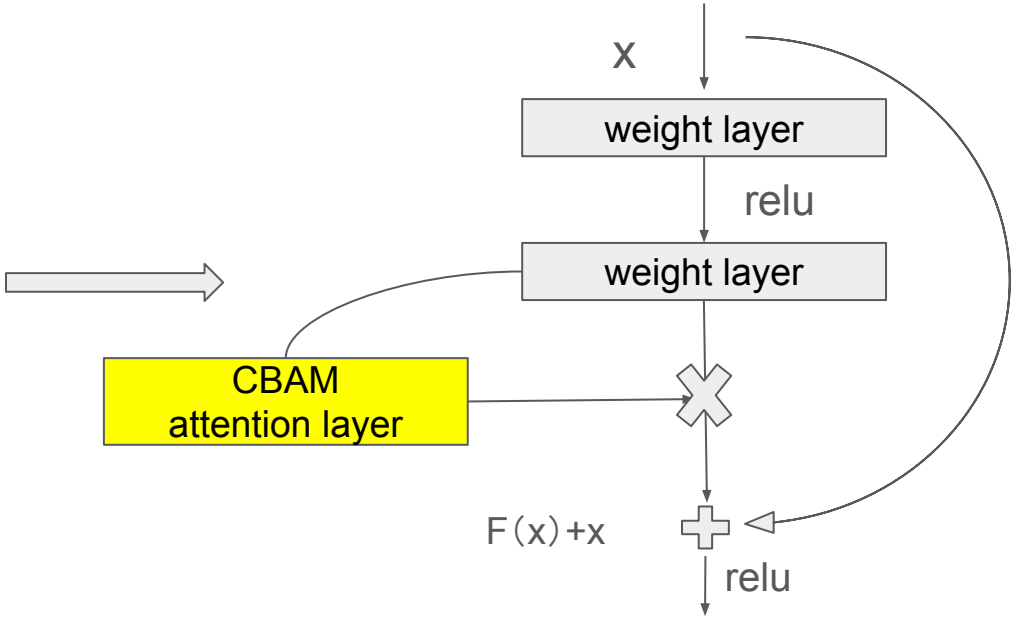
CBAM: Convolutional Block Attention Module



ResNet CBAM

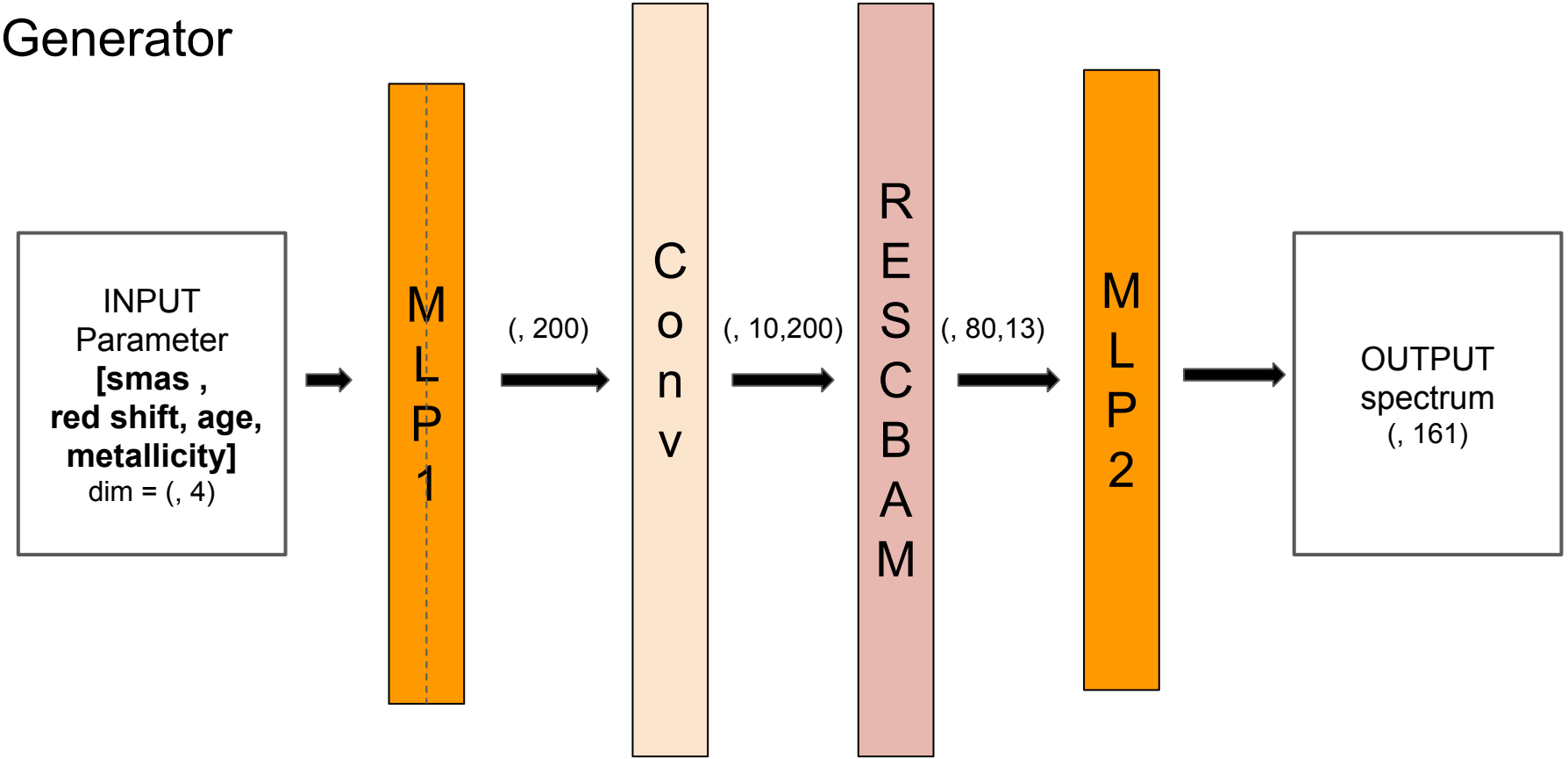
Residual + CBAM block





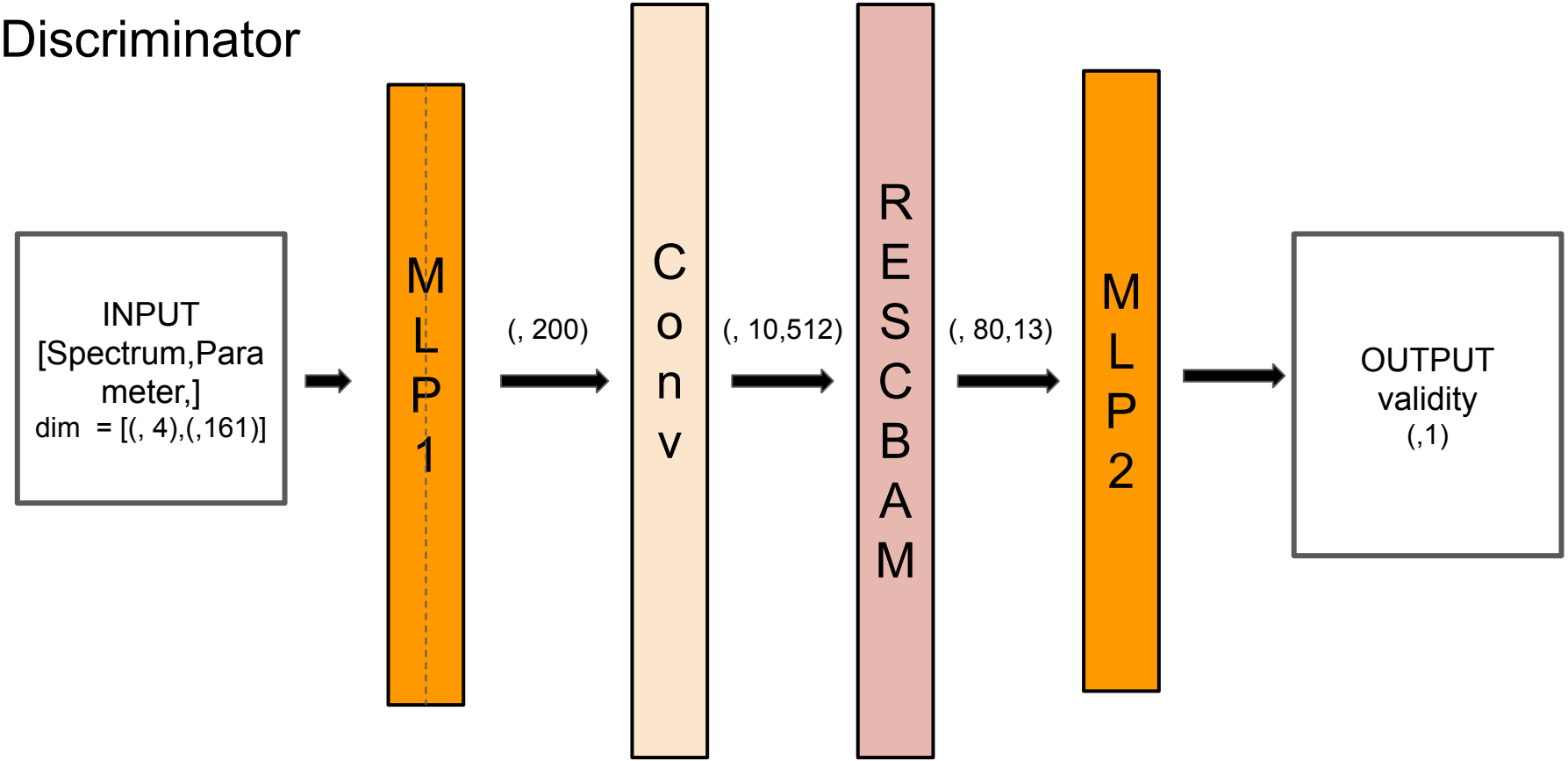
Architecture

Generator



Architecture

Discriminator

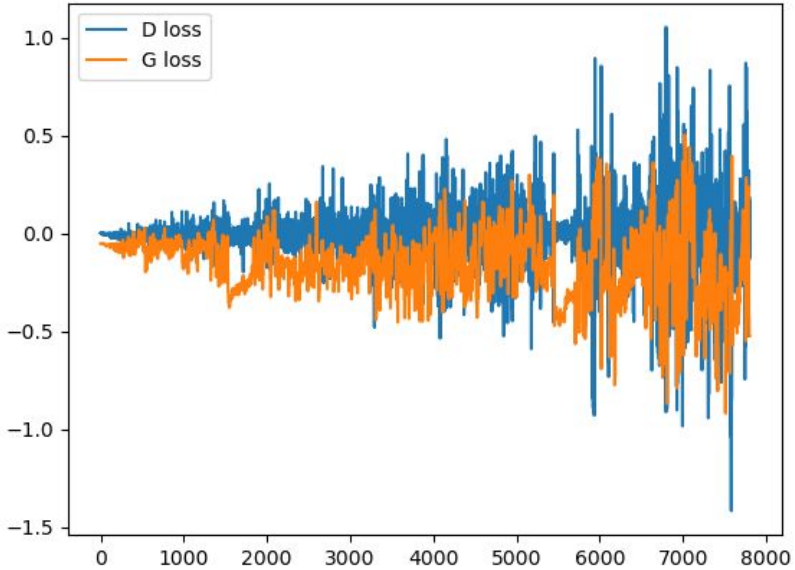


Training

Loss Function :
Wasserstein Lodd

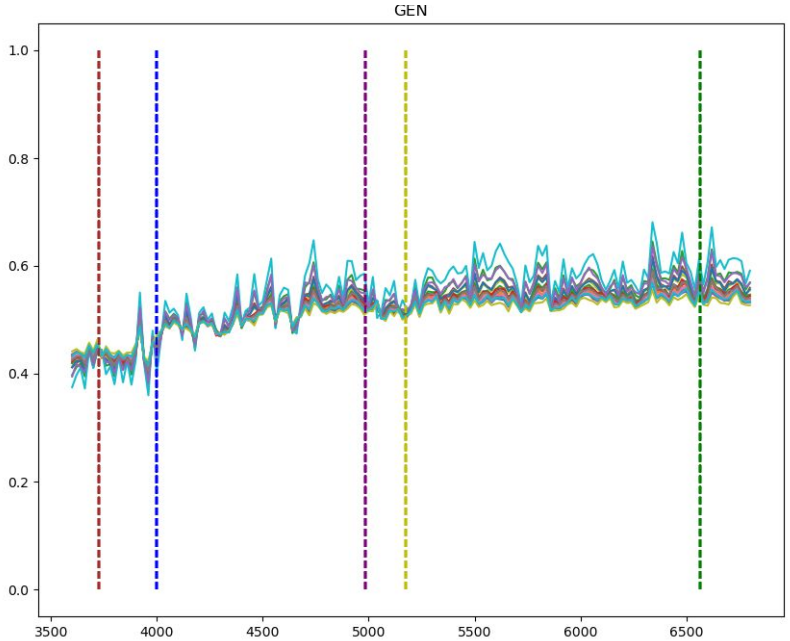
Loss of discriminator :
-torch.mean(dis(real))+torch.mean(dis(fake))

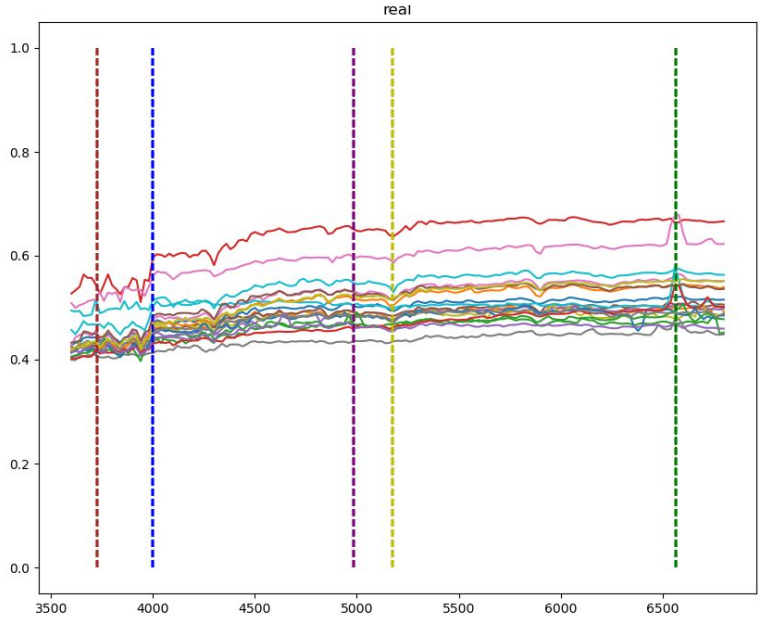
Loss of Generator :
-torch.mean(dis(fake))



Predict

EPOCH = 20 , BATCH SIZE = 20, Clipper boundary (-0.03,0.03)

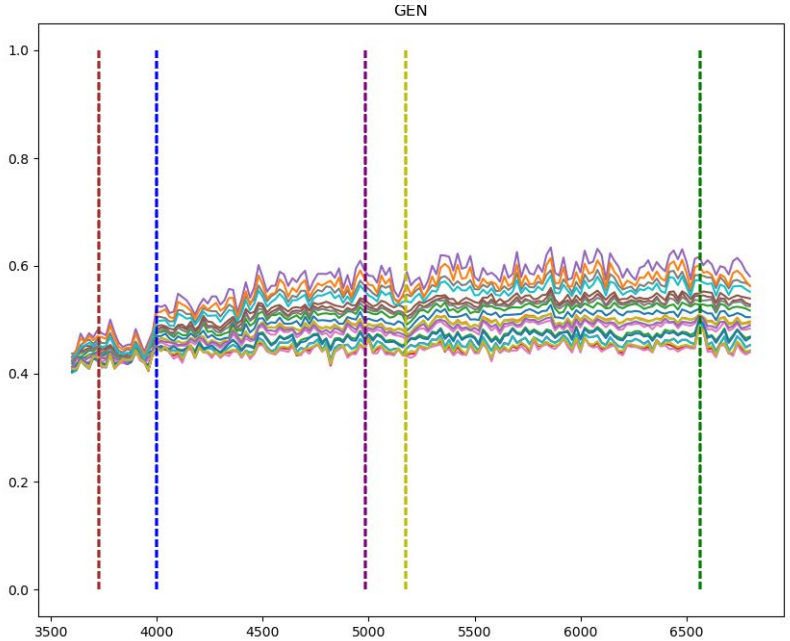


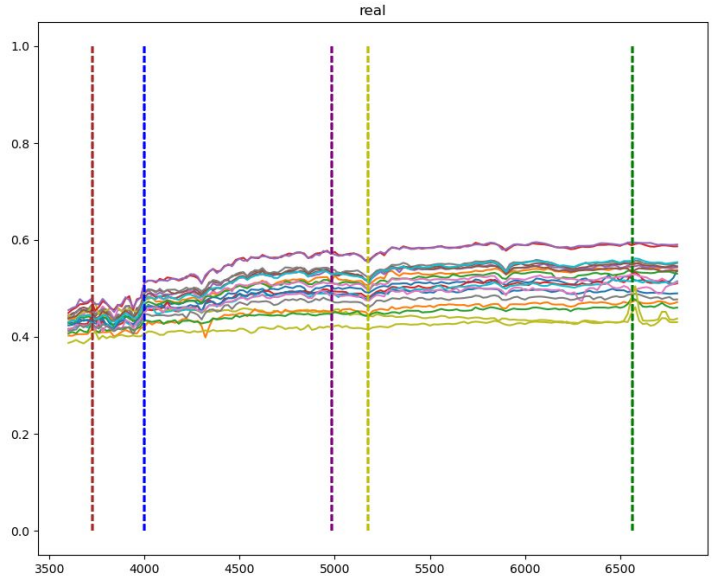


Random sample 10 group parameters

Predict

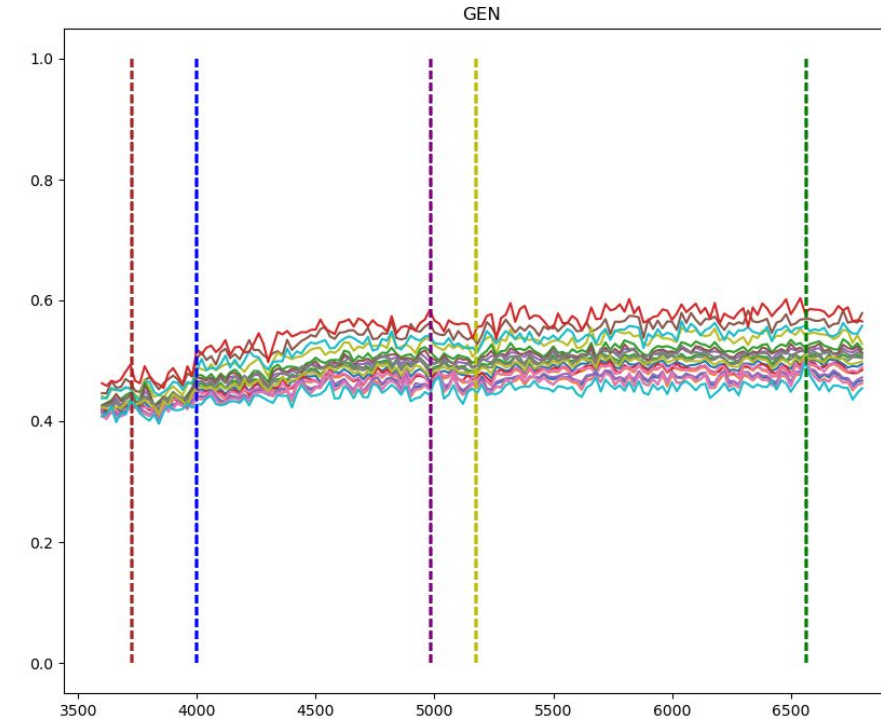
EPOCH = 50 , BATCH SIZE = 50, Clipper boundary (-0.03,0.03)

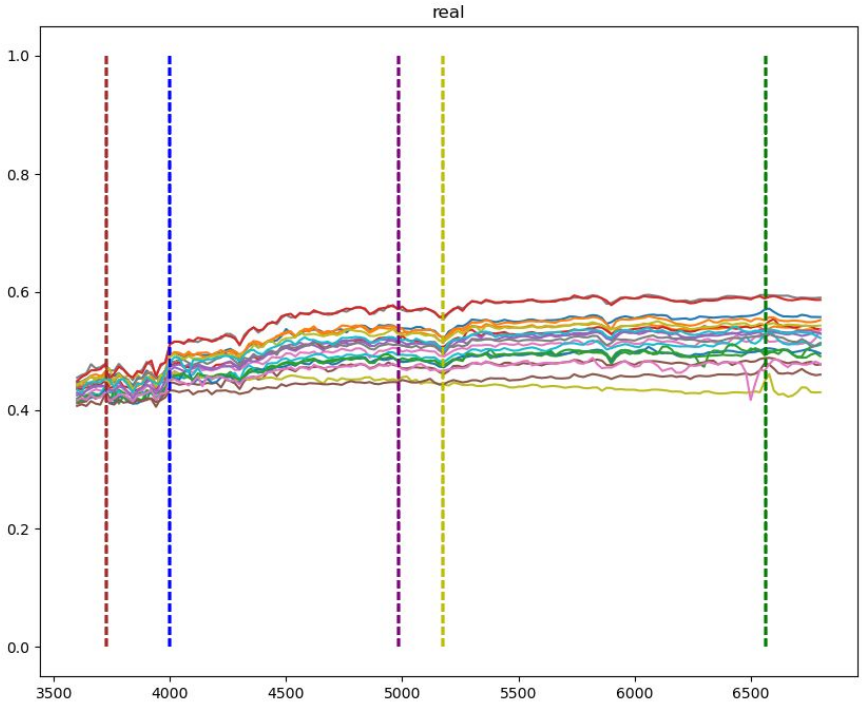




Random sample 10 group parameters

Predict



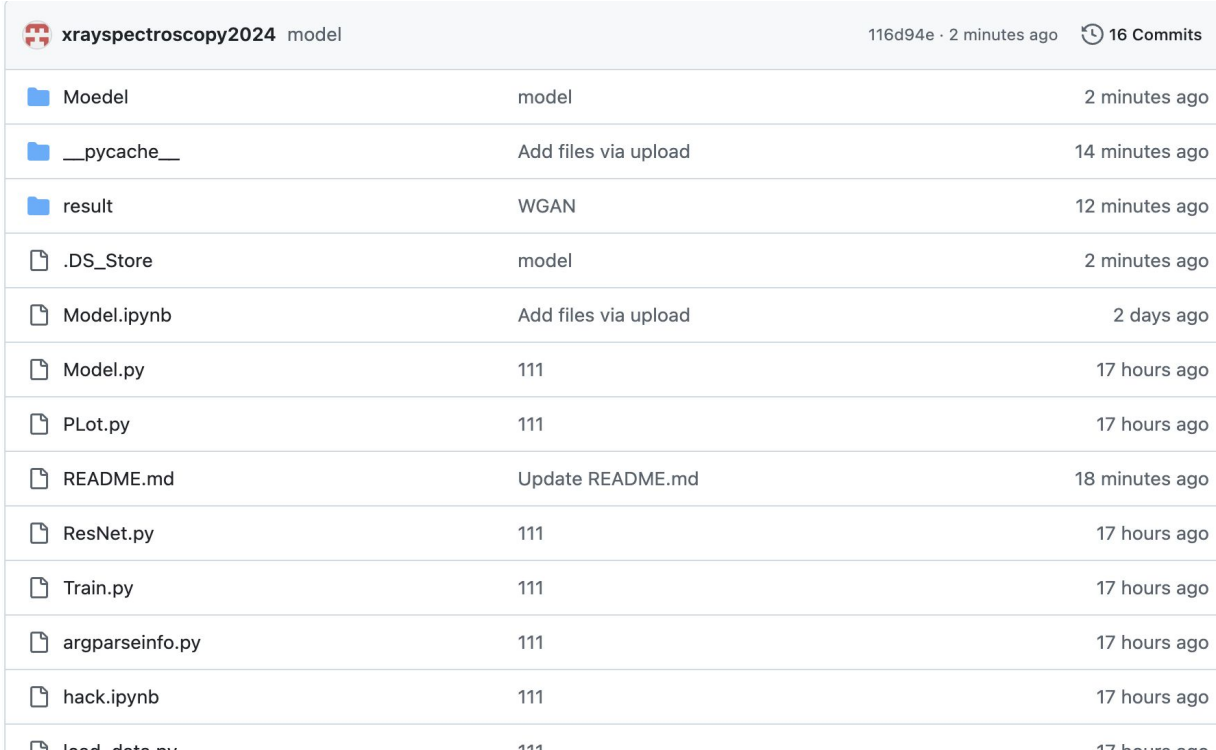


Random sample 10 group parameters

Usage

Model is available on Github :
https://github.com/Astroyimin/WCGAN-for-Galaxy-Spectrum.git

The structure and training need more improvement



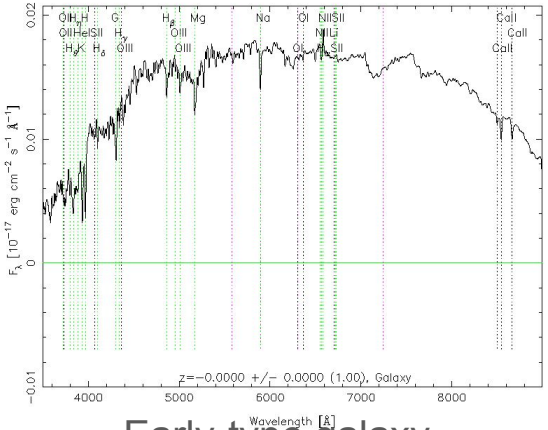
xrayspectroscopy2024 model	116d94e · 2 minutes ago	16 Commits
Moedel	model	2 minutes ago
__pycache__	Add files via upload	14 minutes ago
result	WGAN	12 minutes ago
.DS_Store	model	2 minutes ago
Model.ipynb	Add files via upload	2 days ago
Model.py	111	17 hours ago
PLot.py	111	17 hours ago
README.md	Update README.md	18 minutes ago
ResNet.py	111	17 hours ago
Train.py	111	17 hours ago
argparseinfo.py	111	17 hours ago
hack.ipynb	111	17 hours ago

05. Validation Plan

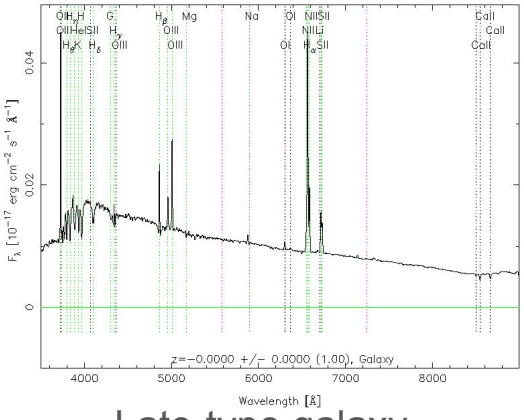
SDSS Spectral Correlation Templates

- Spectral templates for redshift measurement using cross-correlation (https://classic.sdss.org/dr5/algorithms/spectemplates/)
  - Standard form of spectra (33 types)
  - Containing 6 types of galaxy spectra

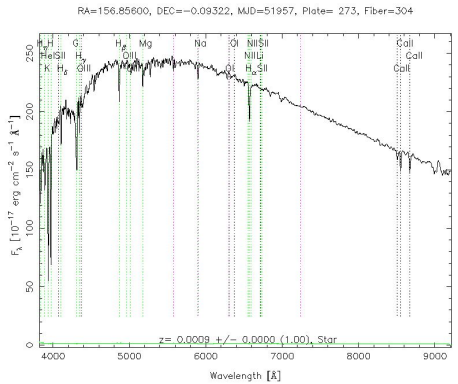
    ⇒ We can check our spectra shape is **similar to galaxy spectra**



Early-type galaxy



Late-type galaxy

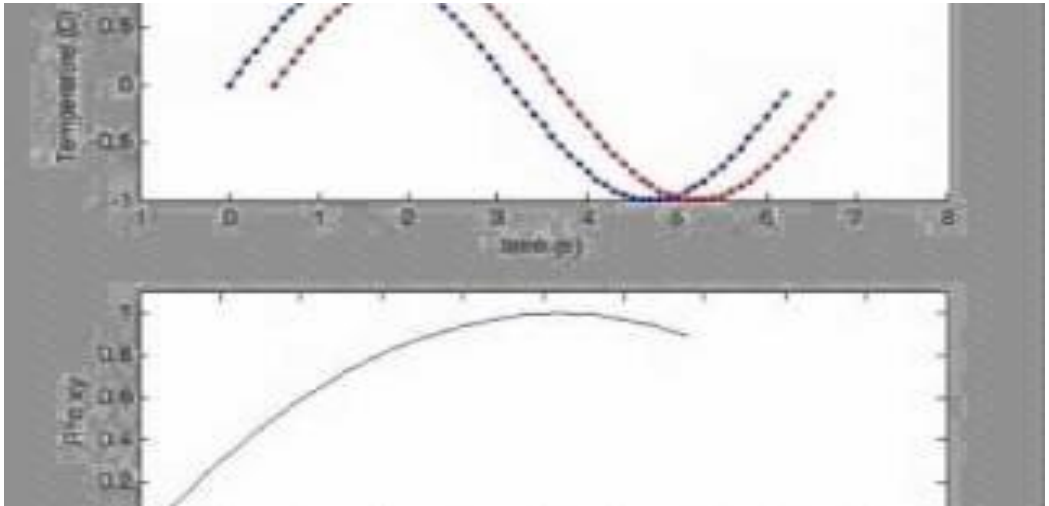


G-star

Cross-Correlation

- A way to measure **how similar two signals** are to each other
  - It checks the similarity after **shifting** one of the series
    → Find the **best-matching shift** and **template**
    → redshift measurement

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} \overline{f(t-\tau)}\, g(t)\, dt$$



Cross-Correlation Tools : RVSNUpy

- Python packages to obtain redshift of spectra using cross-correlation (developed by Taewan Kim, SNU, 2024 in prep.)
  - Provide redshift, best-matching template
  - Provide r-value (cross-correlation coefficient, **how similar signal is?**)

- We can test our GAN model result to answer following questions
  - “Is the model’s output similar to the galaxy’s spectra?”
  - “Is it more similar than the real SDSS spectra (input data)?”
  - “How similar is it quantitatively?”

Q & A

B

Hack presentation:

(4) Interpretation of GAN's latent space/ Improvement of GAN’s performance based the latent space configuration

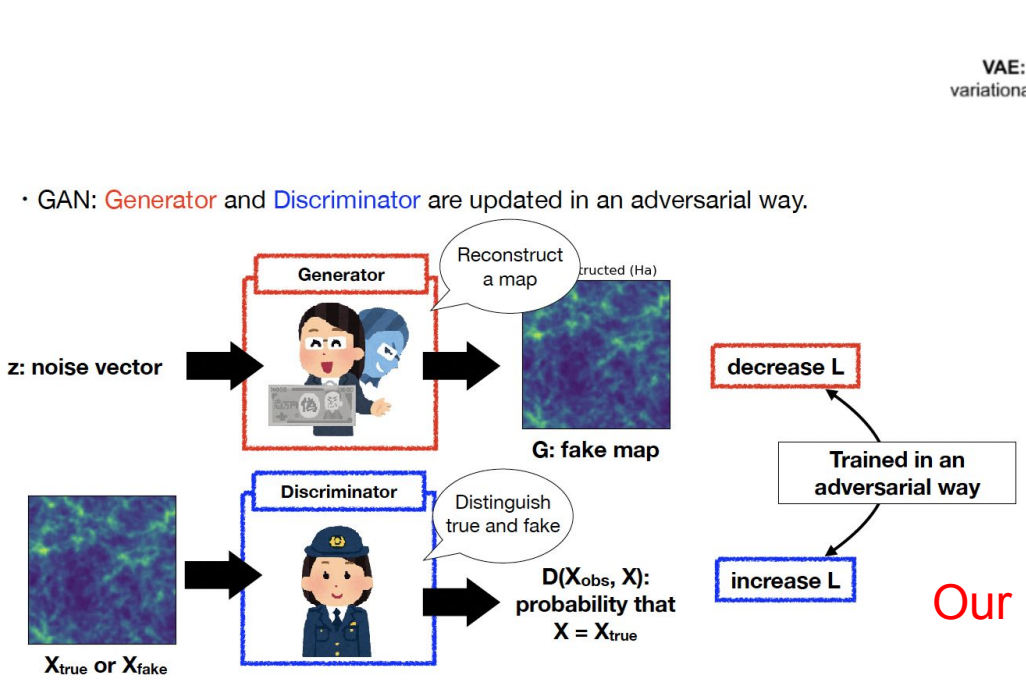
Group B

A3Net Summer School 2024

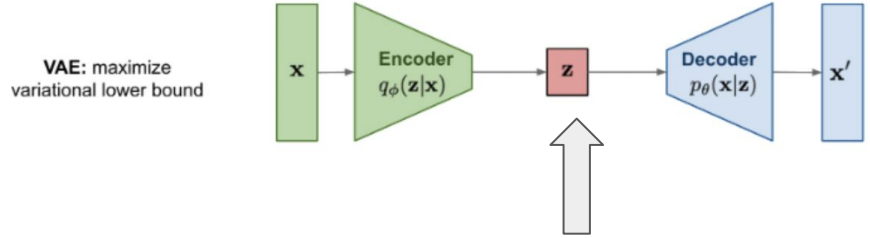
Our goals

- To interpret GAN’s latent space
- To gain a deeper understanding of the latent space of GANs

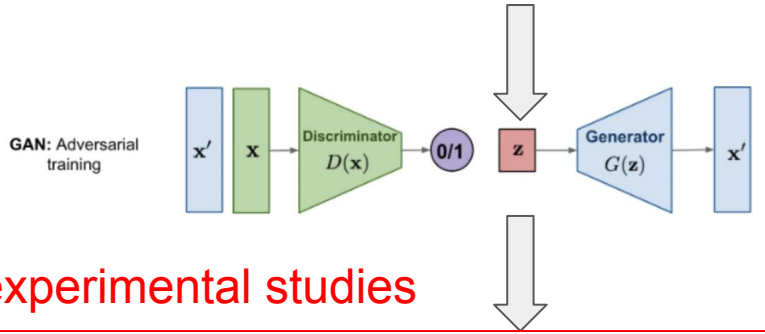
Generative Adversarial Network (GAN)



From Moriwaki-san's slides



Latent Space



Our experimental studies

Physical parameters
Noise + Physical parameters
Noise

Tasks

Obtaining a model with good performance:

- Experimenting different setups for GAN

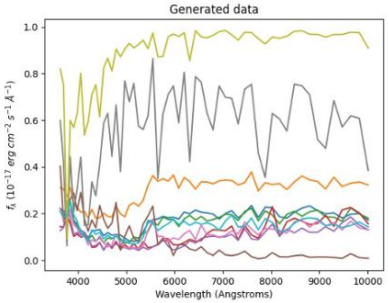
Configuring latent space:

- Experimenting different setups for latent space

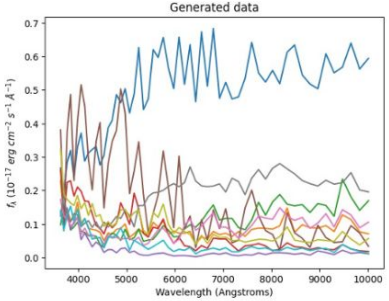
Experimenting different setups for GAN: latent_dim, hidden_dim

1. hidden_dim = 512

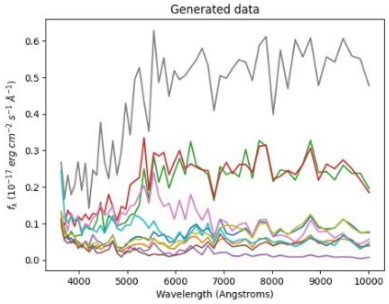
Latent_dim = 5



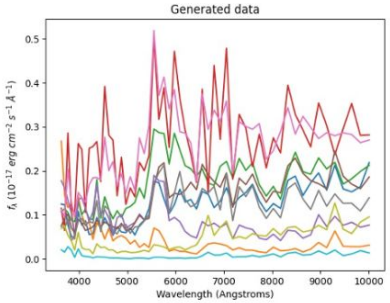
Latent_dim = 10



Latent_dim = 16

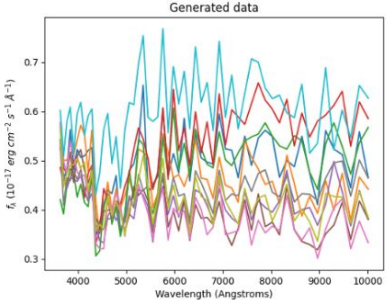


Latent_dim = 25

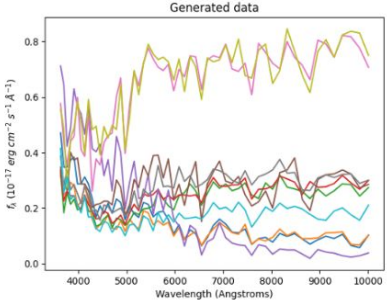


2. latent_dim = 16

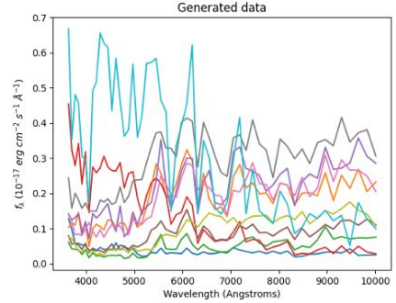
Hidden_dim = 100



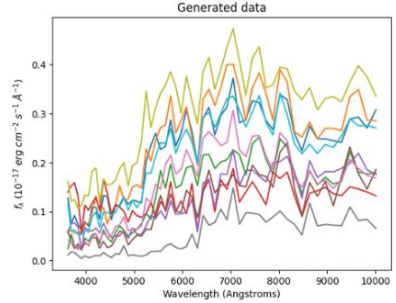
Hidden_dim = 300



Hidden_dim = 500



Hidden_dim = 700

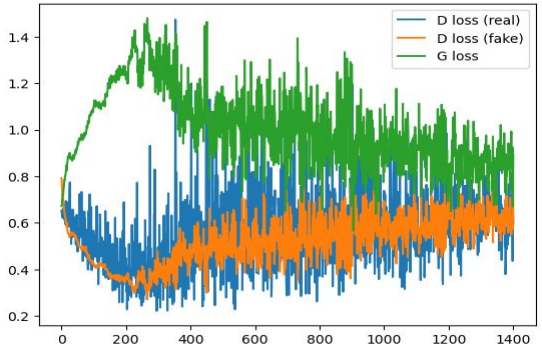


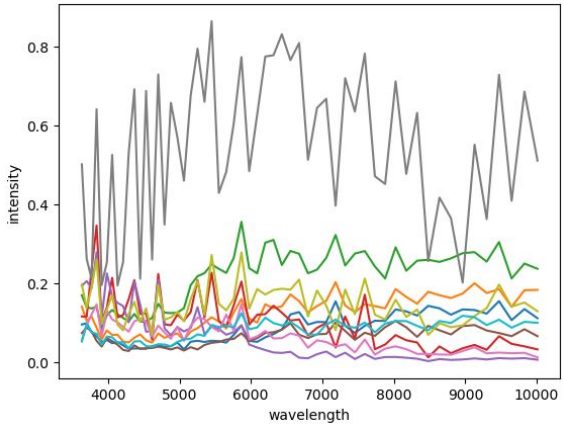
Methods to setting up GAN's latent space

(1) Brute-force approach
- Interpolated noises between two random noises for the latent space
- Changing input noise type

(2) Conditional modelling approach
- Physical parameter as the latent space
- Integrating physical parameters with noise parameters for the latent space

Gaussian noise vs Uniform noise



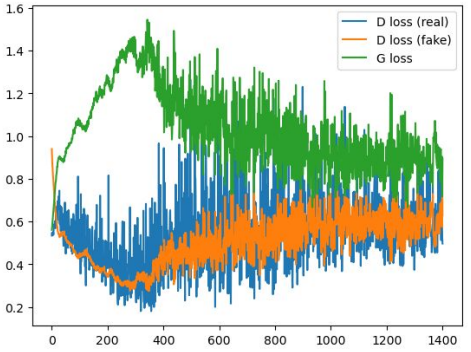


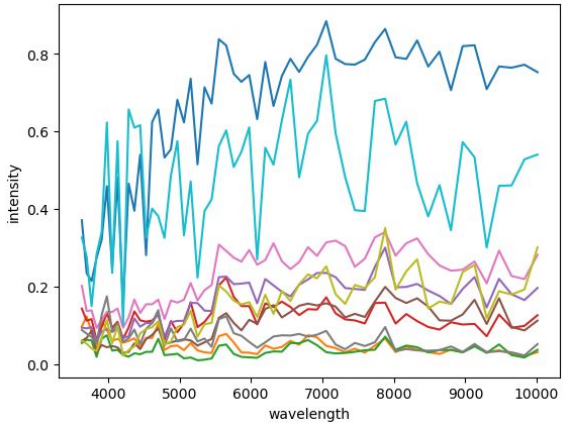
Latent_dim = 16

Hidden_dim = 512

Epochs = 5

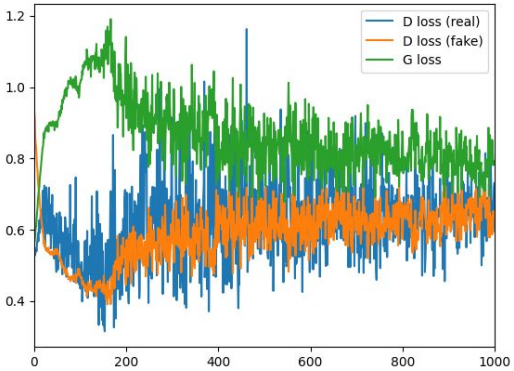
Qualitatively not much difference to be seen.

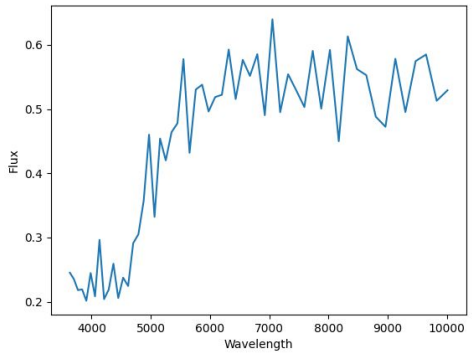




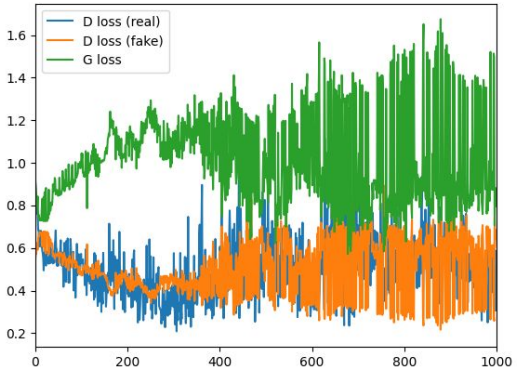
Physical parameters as the latent space (latent_dim=4)

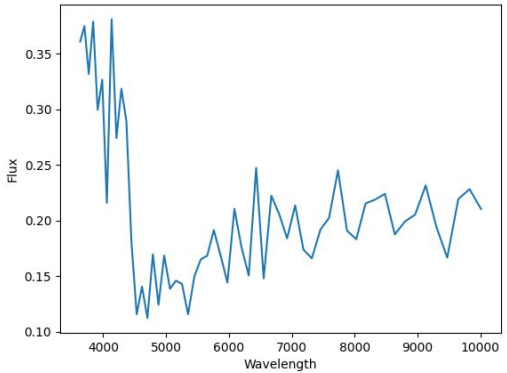
Noise case



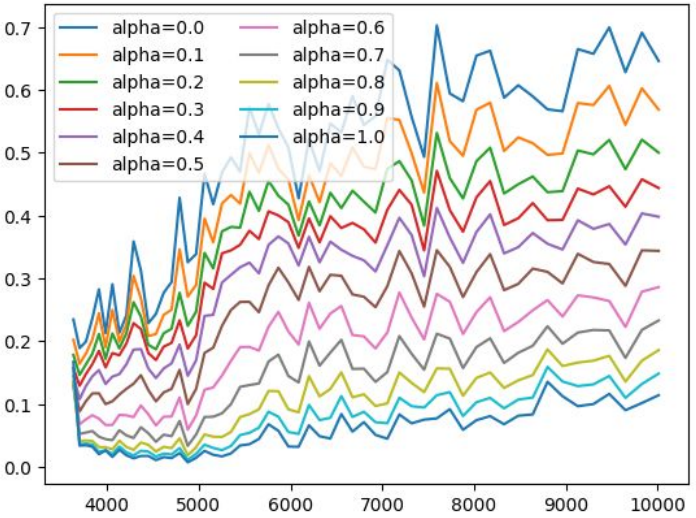


Physical parameter



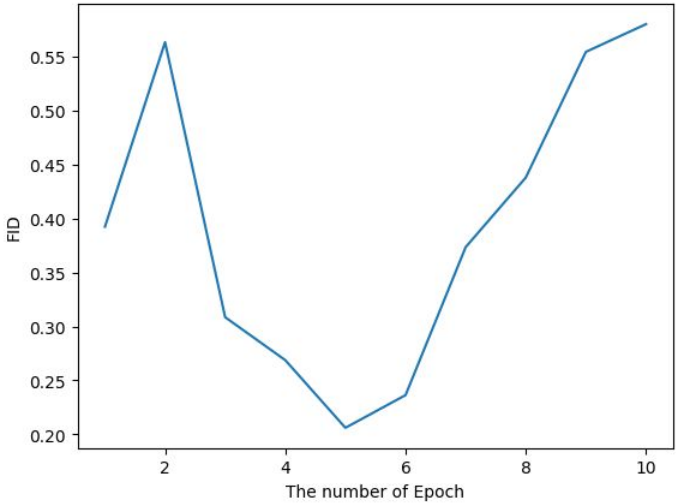


1. Brute-force




```
noise_interp = (1 - alpha) * noise1
+ alpha * noise2
```

Fréchet Inception Distance(FID)
: a metric for evaluating the quality of generated images and specifically developed to evaluate the performance of GAN.

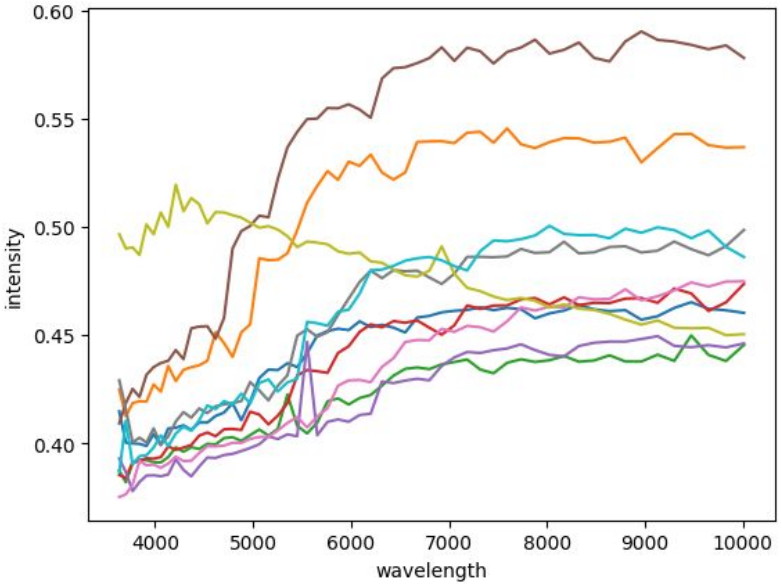


5 is good:)

2. Conditional modelling:

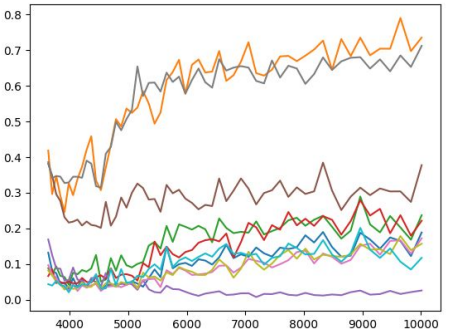
Parameter space as the latent space

Samples from **True** data



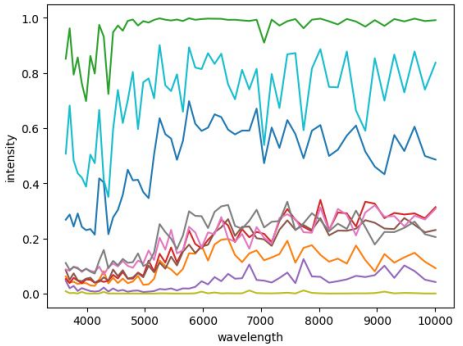
4 latent params:

- 4 random noise



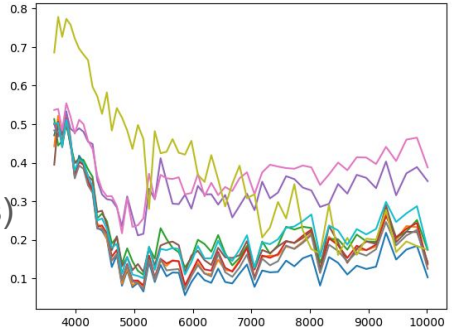
4 latent params:

- 2 random noise
- 2 physical param
  (z, age)



4 latent params:

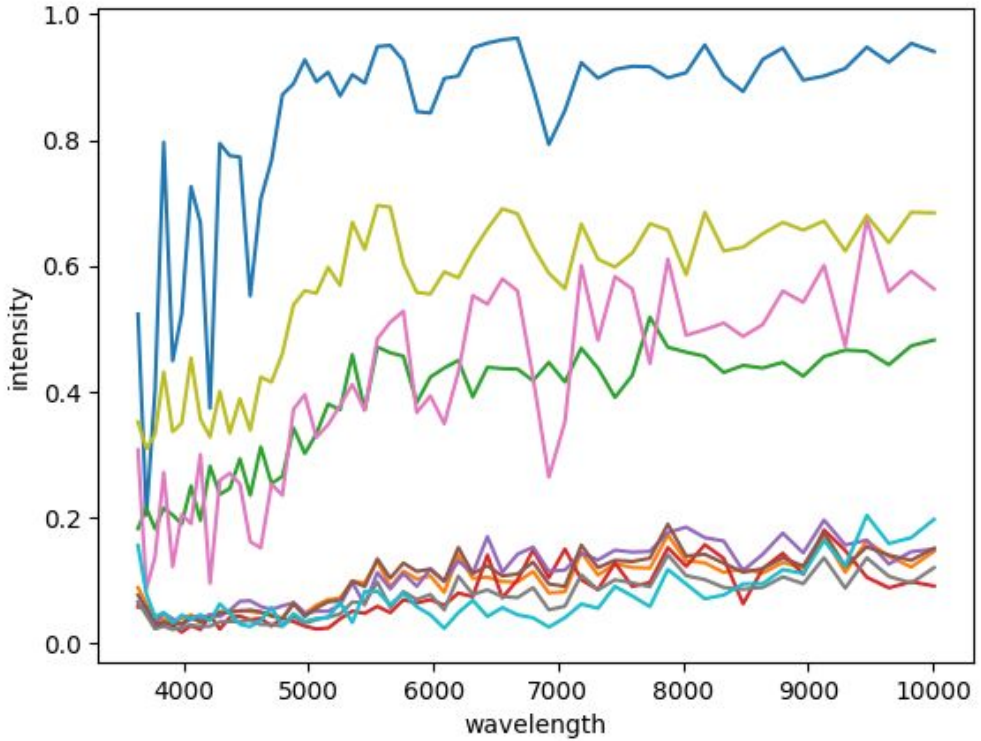
- 4 physical params
  (z, age, metallicity, smass)



8 latent params:

- 4 noise
- 4 physical params

  (z, age, metallicity, smass)



Summary

- We have experimented different setups of GAN model to generate SDSS spectra
- Using physical parameters in the latent space (conditional GAN) did not show any bett

The motivation of this task was to find a way to interpret the latent space of a GAN. Can we find a way to make better choices for choosing the latent space of a GAN?

In conclusion, we can qualitatively see that the latent space is important but it will require much more detailed study to make quantitative statements.

C

Predicting Galaxy redshifts with ML

Member

DONG Shihao, FUJIWARA Tatsuki, JEON Mingyu, KISHIKAWA Ryo, MANZONI Giorgio, NISHIMOTO Shimpei, and TAKAKURA Satoru

Thanks: Michelle Ntampaka & Claire Murray

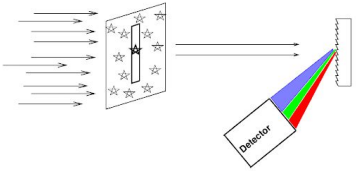
Credit: NASA, ESA, ADAM G. RIESS (STSCI, JHU)

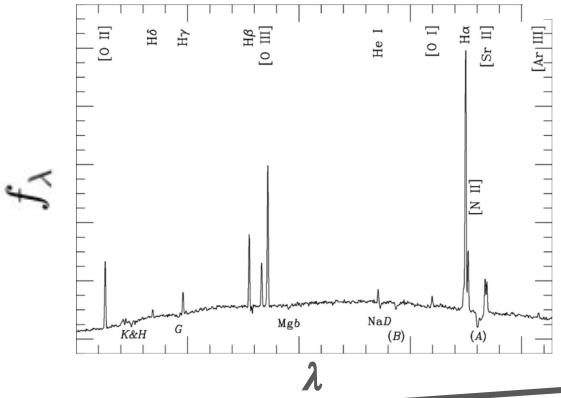
Agenda

- **Introduction and aim of this work**
  - Show Skelton et al. 2014 as previous work
- **Results**
  - **Show results of ML models**
    - Gradient-Boosted Tree
    - Random Forest
    - Multi-Layer Perceptron Regressor
    - Support Vector Regression
  - **Importance of Feature Engineering**
- **Conclusion**

Introduction and aim of this work

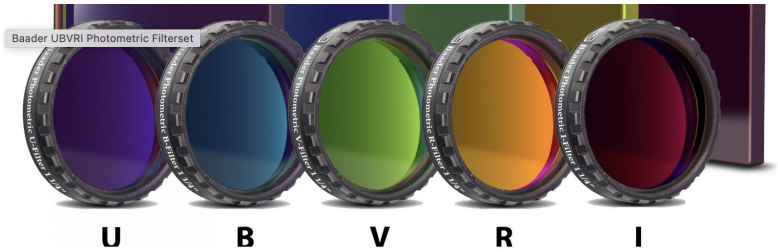
Spectroscopic redshift (spec-z)





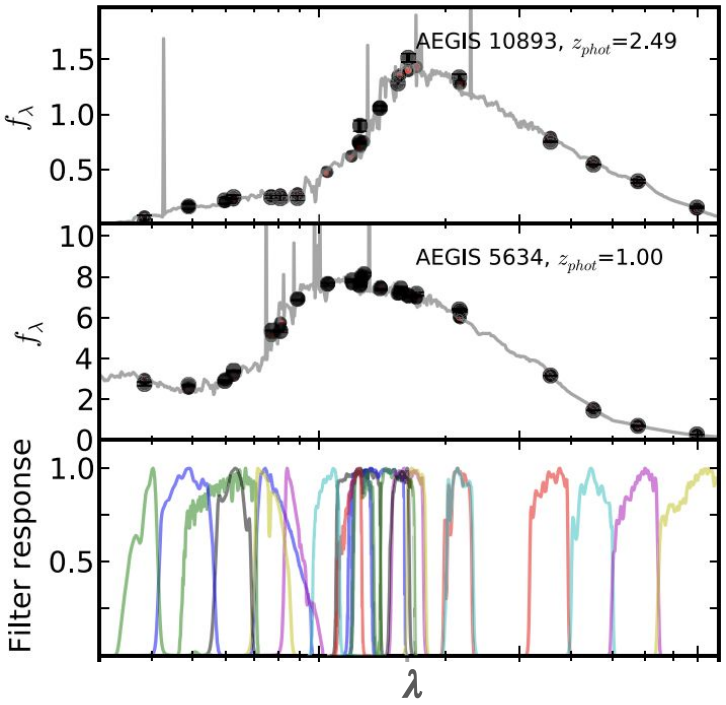
Observing Time (money)

Photometric redshift (photo-z)



Traditionally to estimate galaxy redshifts from photometric bands we rely on **SED fitting**

Introduction and aim of this work



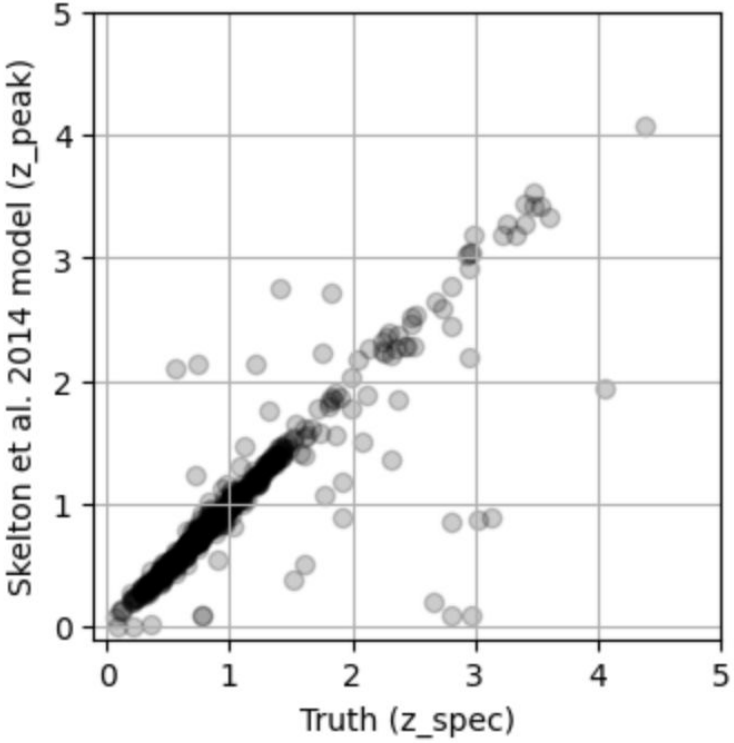
The disadvantage of SED fitting

1. **Need for the library** of theoretical models
2. Depends on the **assumption on star formation history** of the galaxy

The more filter we have
the better the SED is constrained

AIM **CAN ML DO BETTER?**

The Data (Skelton et al. 2014)



1. CANDLES+3D HST (HST) collected both **photo-z** and **spec-z** over **900 arcmin^2**
2. Using **9 HST filter**
3. They apply EAZY photo-z code (**traditional** template based photo-z)

AIM **CAN ML DO BETTER?**

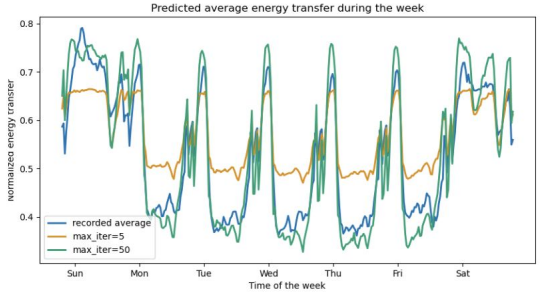
Machine learning models we tested

- Random Forest
- Gradient-Boosted Tree
- Multi-Layer Perceptron Regressor
- Support Vector Regression
- ...

Regression

Predicting a continuous-valued attribute associated with an object.

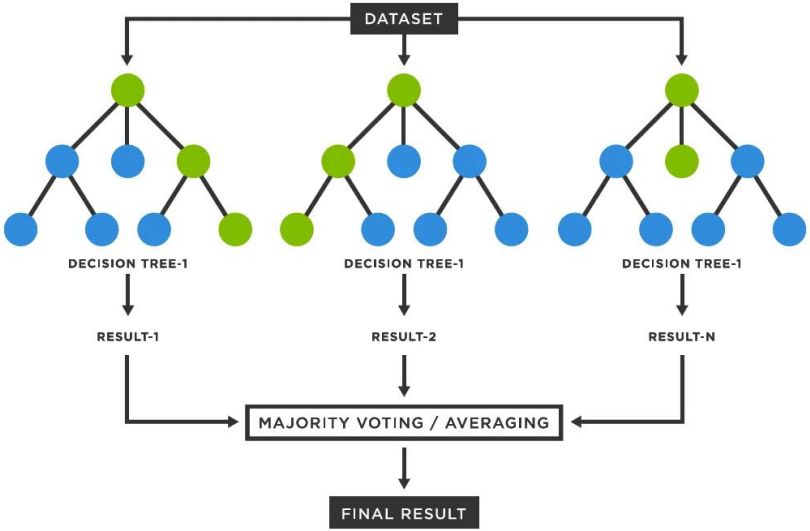
Applications: Drug response, stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...



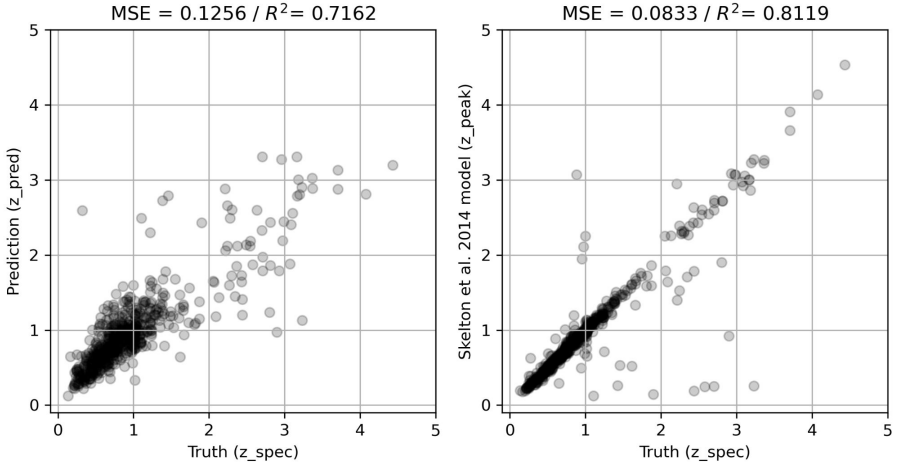
Glossary

MSE	MSE is a common metric used to evaluate the performance of regression models. It measures the average of the squares of the differences between predicted and actual values. We compare mse of each model.

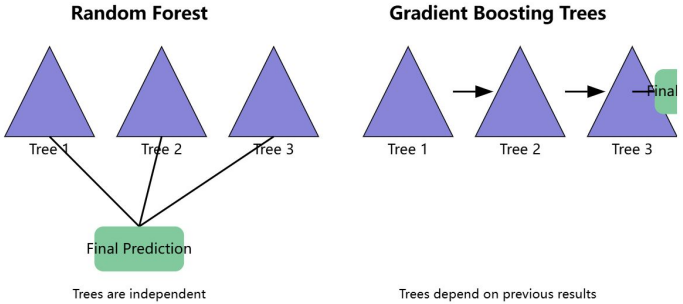
Random Forest



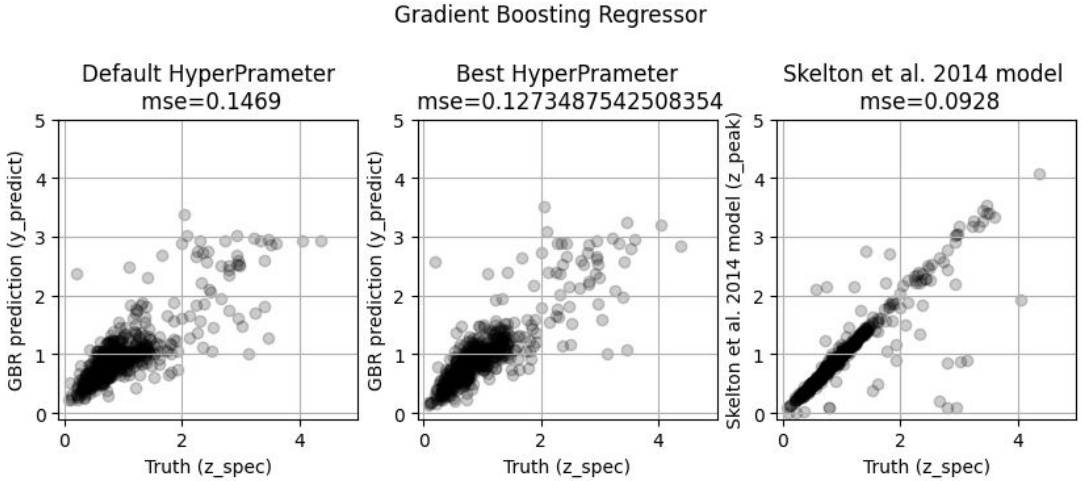
Random Forest vs. Skelton et al. 2014



GBT(Gradient-boosted Tree)



GBT vs. skelton et al.2014

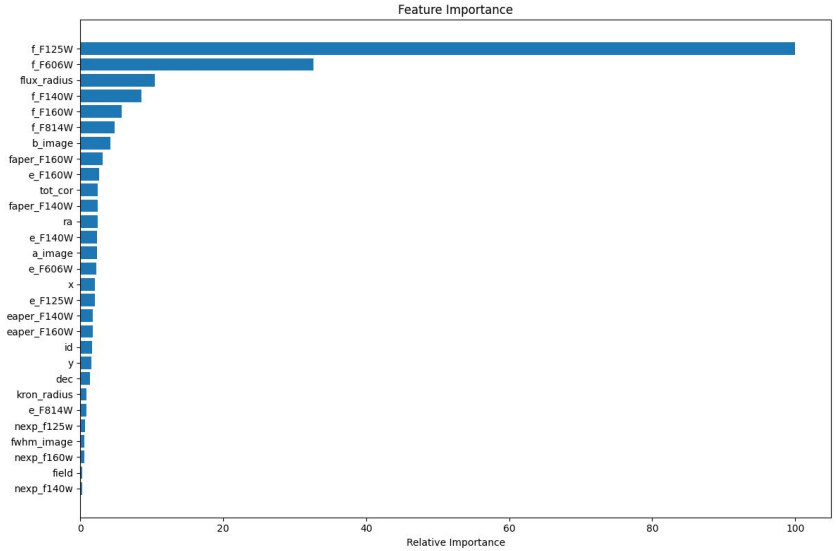


GBT

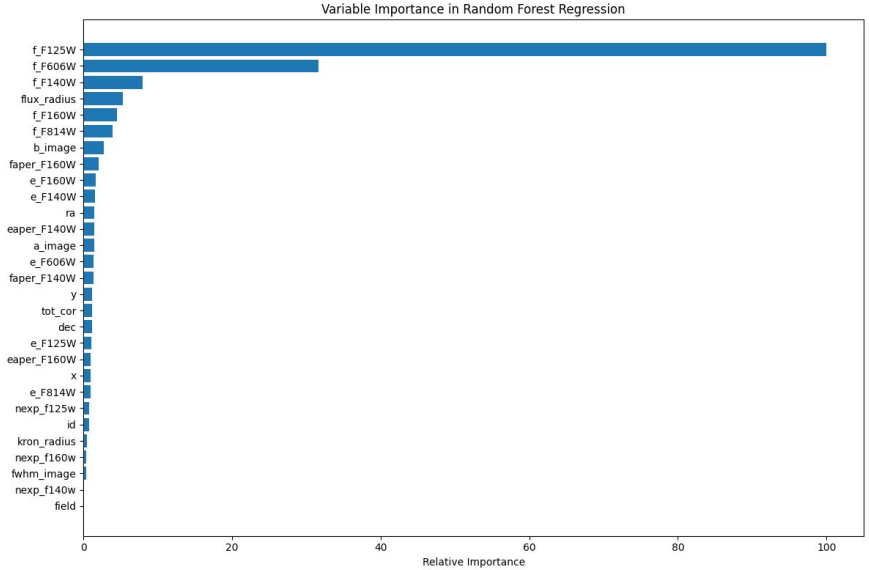
- sensitive to parameter settings
- requires careful tuning
- Time consuming…

We searched hyperparameters at 500 times using bayesian optimization, and picked up the hyperparameter pattern when MSE became the lowest

GBT vs Random Forest



Feature importance in GBT

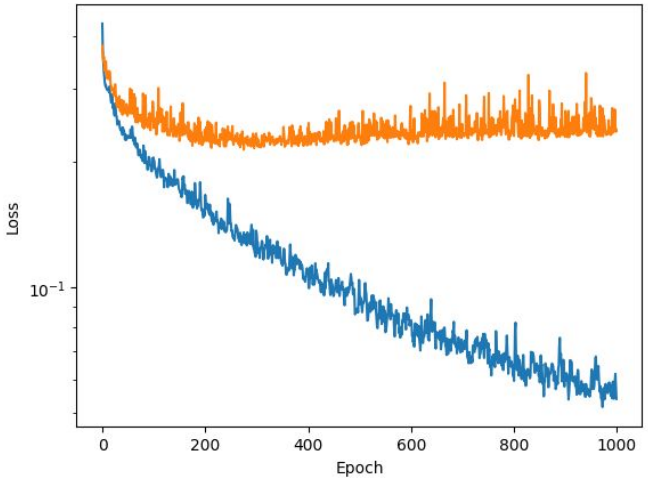


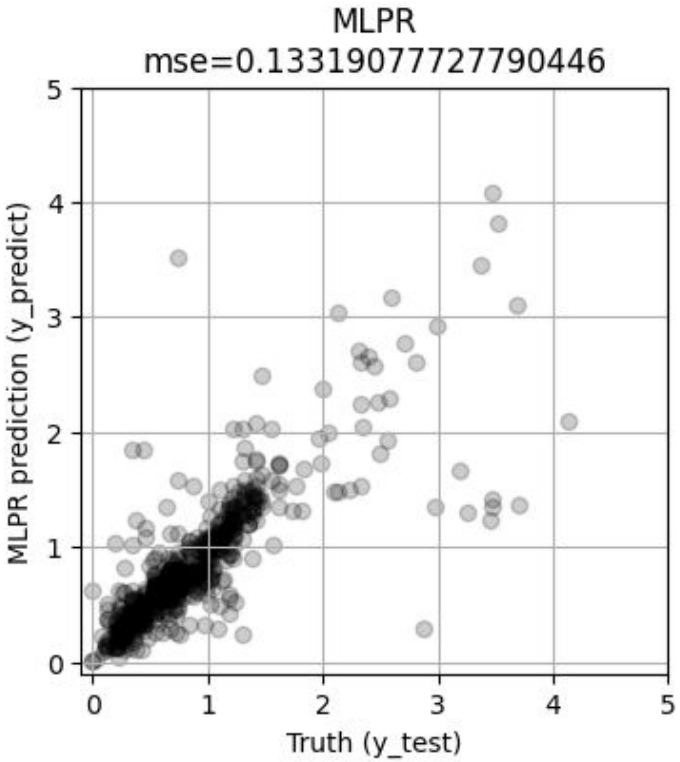
Feature importance in RF

MLP regressor

Hidden layers: (256, 256, 256)

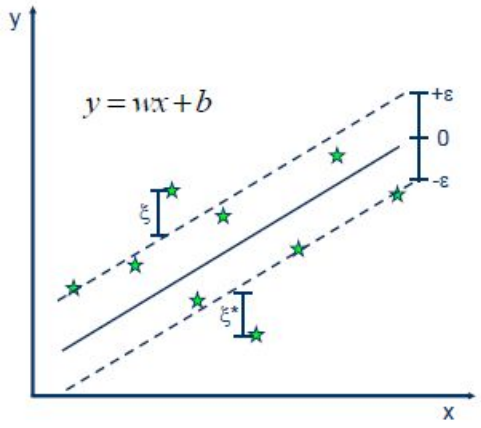
Loss: L1Loss





notebook

Support Vector Regression



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

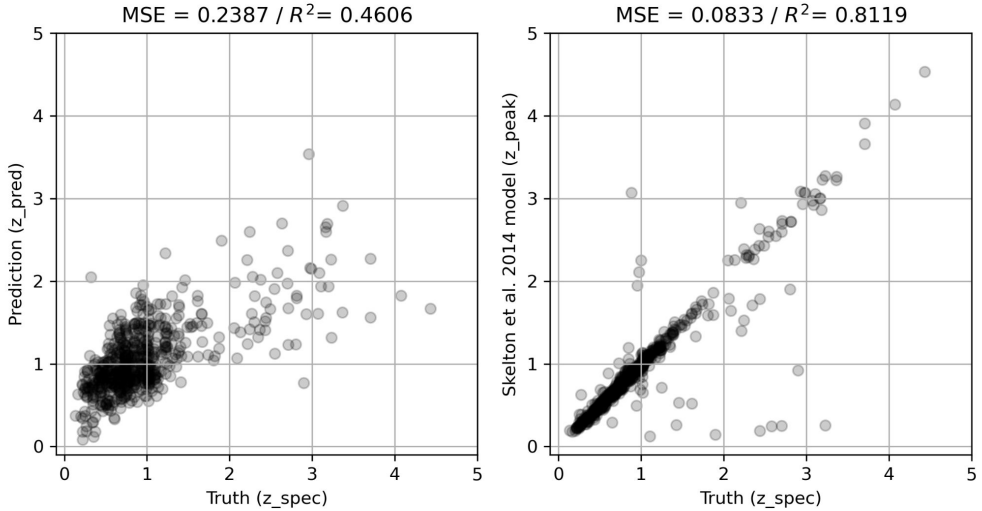
- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

Support Vector Regression vs. Skelton et al. 2014



Regression experiment using PyCaret

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
rf	Random Forest Regressor	0.2238	0.1337	0.3632	0.7124	0.1448	0.2432
lightgbm	Light Gradient Boosting Machine	0.2261	0.1364	0.3679	0.7061	0.1459	0.2478
et	Extra Trees Regressor	0.2410	0.1463	0.3812	0.6839	0.1518	0.2649
gbr	Gradient Boosting Regressor	0.2448	0.1472	0.3824	0.6821	0.1541	0.2775
dt	Decision Tree Regressor	0.2983	0.2440	0.4894	0.4790	0.1943	0.3187
lr	Linear Regression	0.3652	0.2676	0.5166	0.4188	0.2156	0.4513
br	Bayesian Ridge	0.3644	0.2677	0.5167	0.4188	0.2146	0.4483
ridge	Ridge Regression	0.3693	0.2859	0.5343	0.3800	0.2164	0.4496
ada	AdaBoost Regressor	0.4370	0.3005	0.5478	0.3425	0.2463	0.6345
en	Elastic Net	0.4030	0.3662	0.6040	0.2117	0.2437	0.5309
lasso	Lasso Regression	0.4194	0.4001	0.6315	0.1382	0.2570	0.5683
llar	Lasso Least Angle Regression	0.4194	0.4001	0.6315	0.1382	0.2570	0.5683
omp	Orthogonal Matching Pursuit	0.4591	0.4582	0.6757	0.0137	0.2838	0.6489
dummy	Dummy Regressor	0.4618	0.4665	0.6818	-0.0039	0.2868	0.6572
knn	K Neighbors Regressor	0.4838	0.4846	0.6955	-0.0506	0.2937	0.6450
huber	Huber Regressor	0.4544	0.4959	0.7006	-0.0610	0.2860	0.6099
par	Passive Aggressive Regressor	0.9365	1.9822	1.2151	-3.1235	0.4592	1.3608
lar	Least Angle Regression	0.8460	4.7030	1.2703	-10.2415	0.3619	1.4869

Random Forest is best

Random Forest for photometric redshift estimation

Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **505,** 4847–4856 (2021) https://doi.org/10.1093/mnras/stab1513
Advance Access publication 2021 May 29

Benchmarking and scalability of machine-learning methods for photometric redshift estimation

Ben Henghes ,[1]★ Connor Pettitt,[2] Jeyan Thiyagalingam,[2]★ Tony Hey[2] and Ofer Lahav[1]

[1]*Department of Physics & Astronomy, University College London, Gower Street, London WC1E 6BT, UK*
[2]*Scientific Computing Department, Rutherford Appleton Laboratory, Science and Technology Facilities Council (STFC), Harwell Campus, Didcot OX11 0QX, UK*

Accepted 2021 May 14. Received 2021 April 29; in original form 2021 April 5

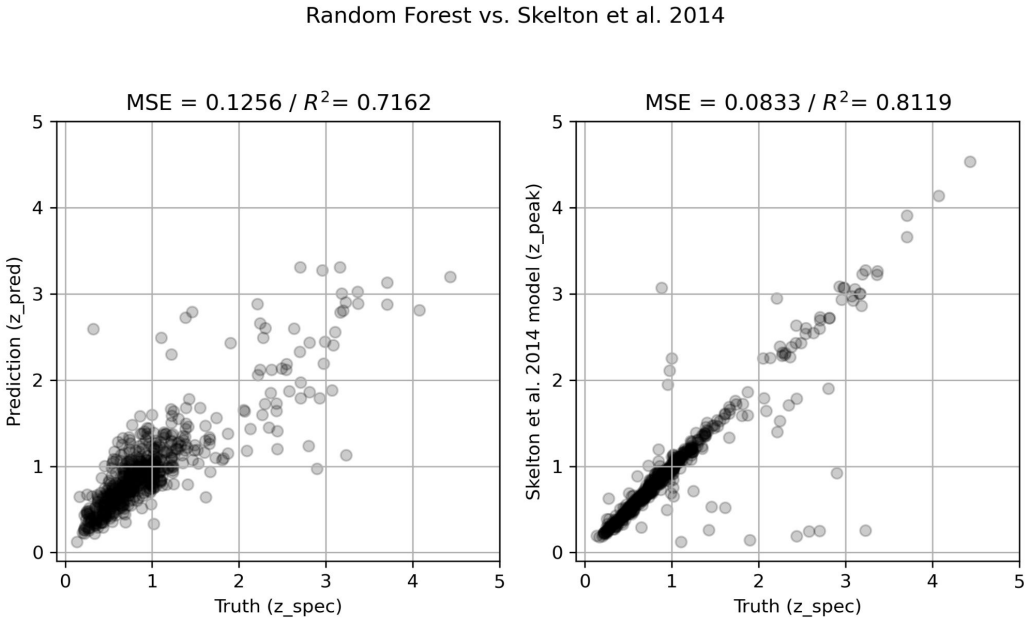
ABSTRACT

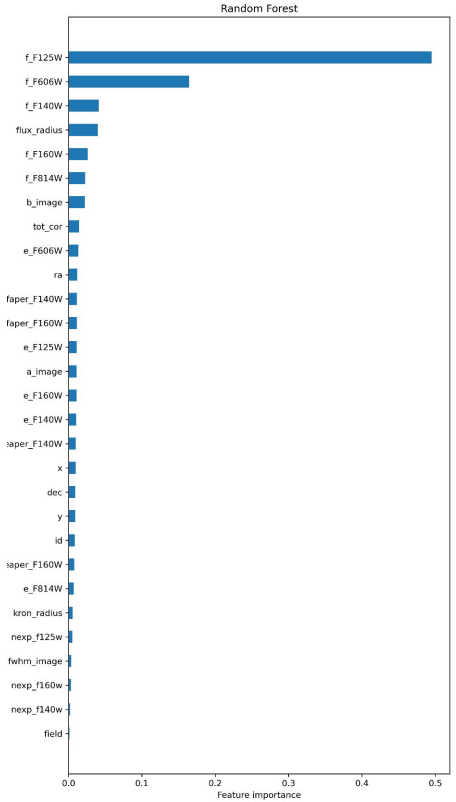
Obtaining accurate photometric redshift (photo-z) estimations is an important aspect of cosmology, remaining a prerequisite of many analyses. In creating novel methods to produce photo-z estimations, there has been a shift towards using machine-learning techniques. However, there has not been as much of a focus on how well different machine-learning methods scale or perform with the ever-increasing amounts of data being produced. Here, we introduce a benchmark designed to analyse the performance and scalability of different supervised machine-learning methods for photo-z estimation. Making use of the Sloan Digital Sky Survey (SDSS – DR12) data set, we analysed a variety of the most used machine-learning algorithms. By scaling the number of galaxies used to train and test the algorithms up to one million, we obtained several metrics demonstrating the algorithms' performance and scalability for this task. Furthermore, by introducing a new optimization method, time-considered optimization, we were able to demonstrate how a small concession of error can allow for a great improvement in efficiency. From the algorithms tested, we found that the Random Forest performed best with a mean squared error, MSE = 0.0042; however, as other algorithms such as Boosted Decision Trees and k-Nearest Neighbours performed very similarly, we used our benchmarks to demonstrate how different algorithms could be superior in different scenarios. We believe that benchmarks like this will become essential with upcoming surveys, such as the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST), which will capture billions of galaxies requiring photometric redshifts.

Key words: methods: data analysis – galaxies: distances and redshifts – cosmology: observations.

Importance of Feature Engineering

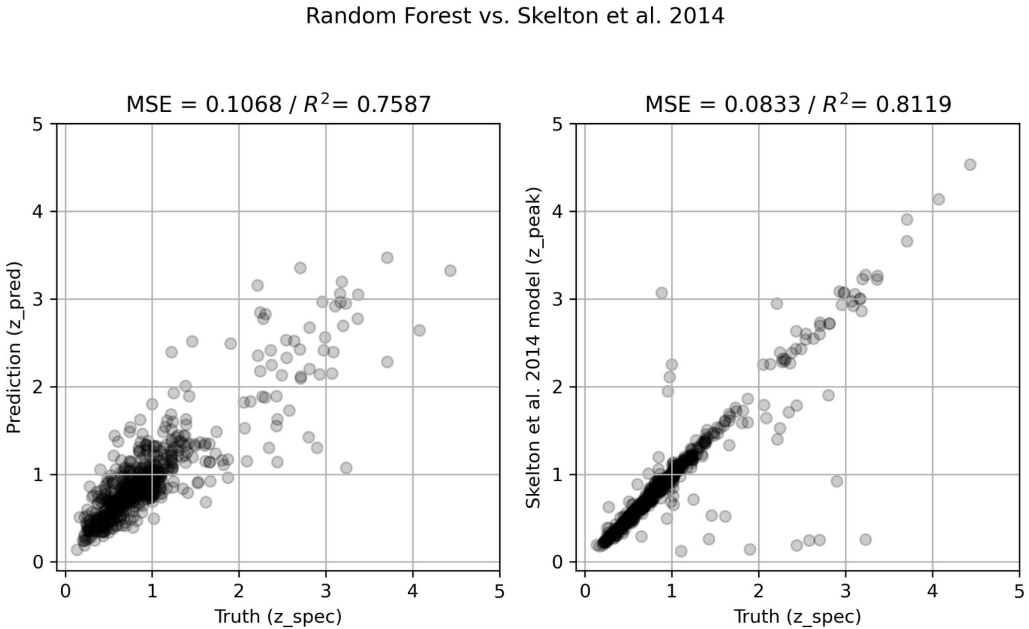
When using almost all 29 features

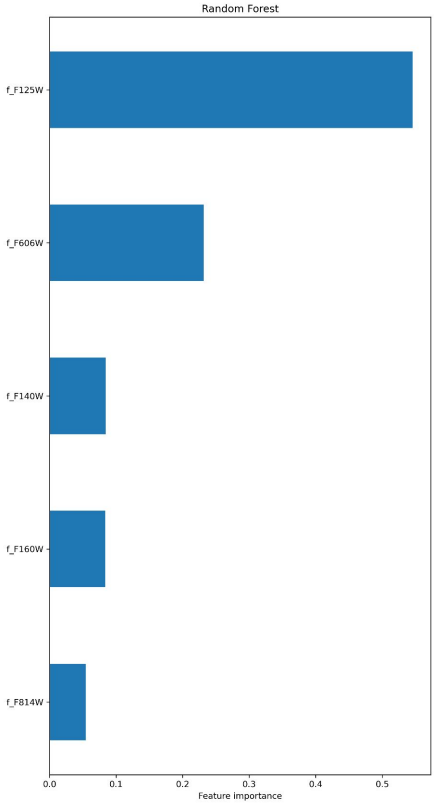




Importance of Feature Engineering

When using only 5 photometric bands

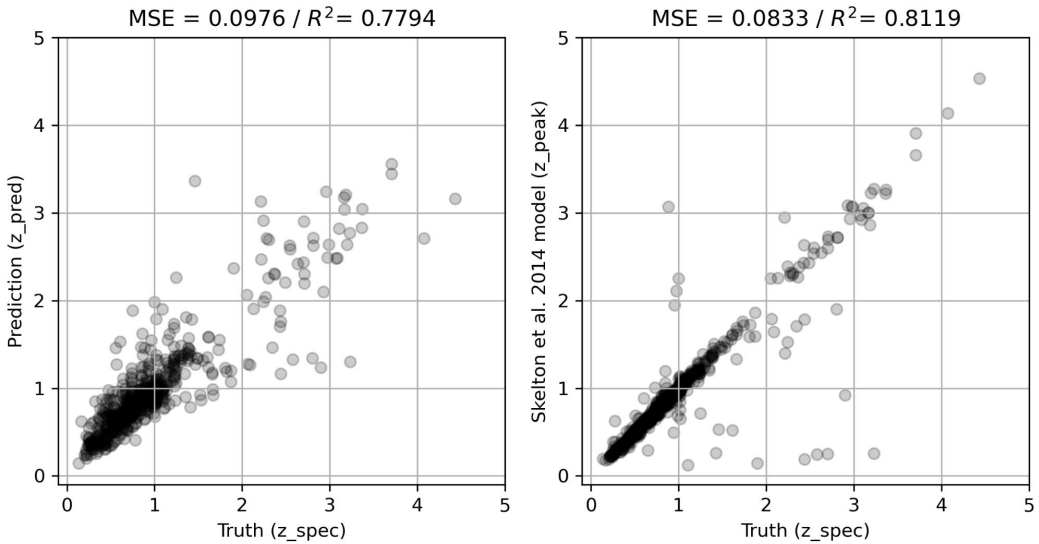


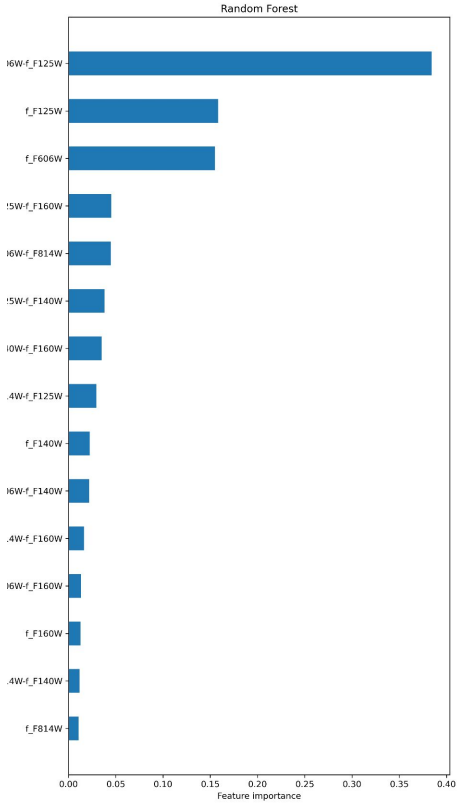


Importance of Feature Engineering

When using only 5 photometric bands
+ all the possible combinations of colors

Random Forest vs. Skelton et al. 2014





Conclusion

We predict photometric redshifts using machine learning models and compare their performance with the traditional method.

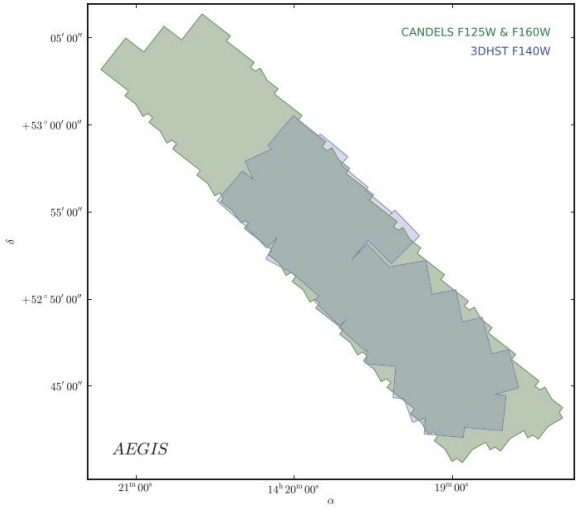
The best machine learning model (random forest) does not outperform the traditional method.

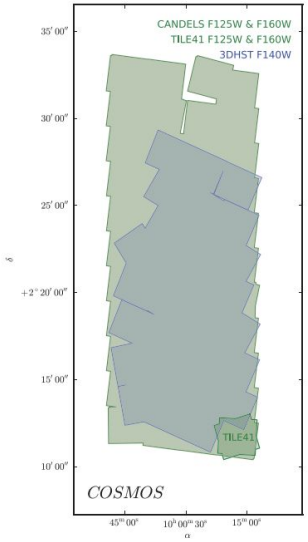
We observe that including irrelevant features can degrade the performance of ML models, while creating physically important features improves their accuracy.

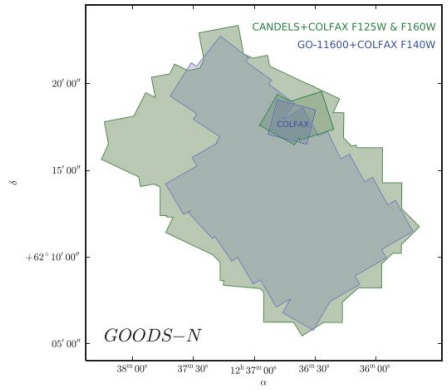
This shows the critical role of feature engineering in machine learning.

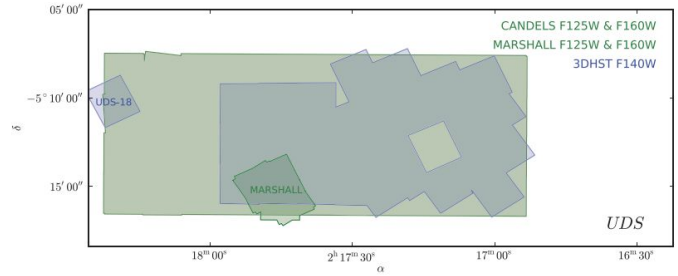
Additional slides for discussion

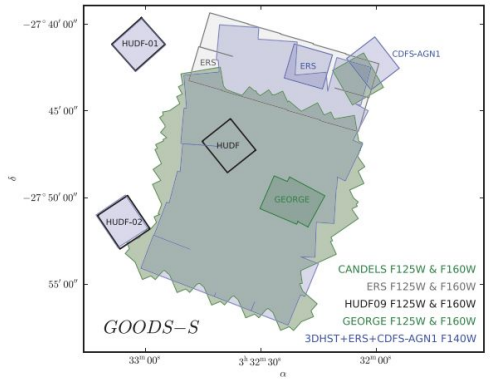
The fields covered by Skelton et al. 2014









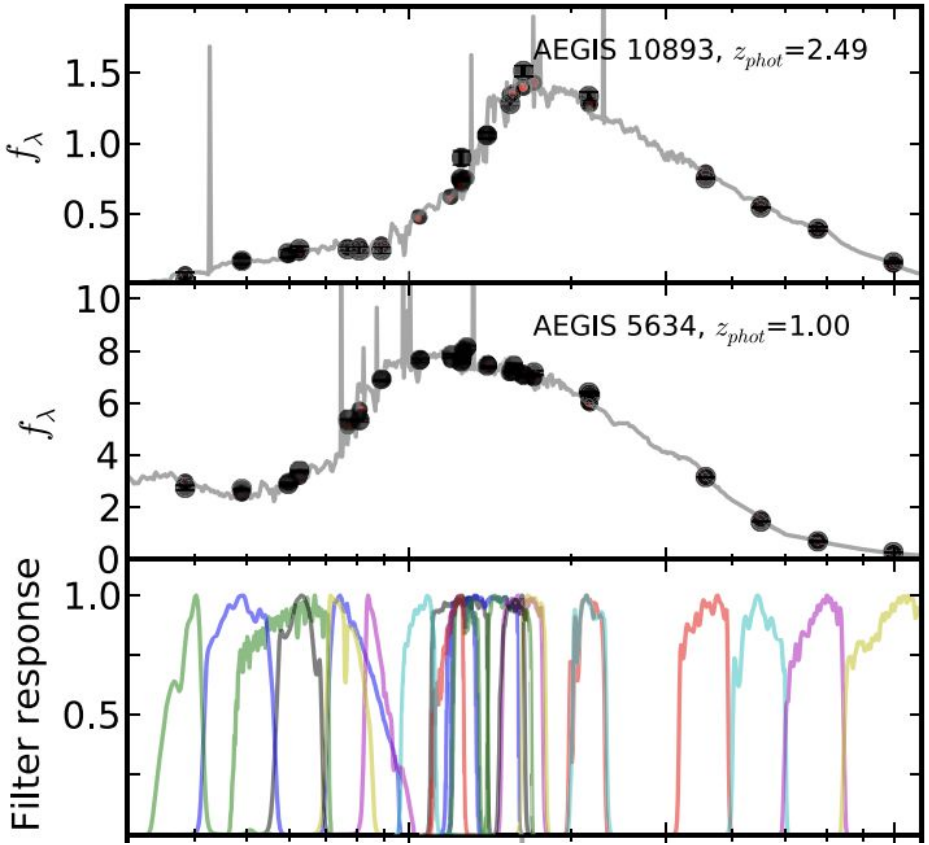


The fields covered by Skelton et al. 2014

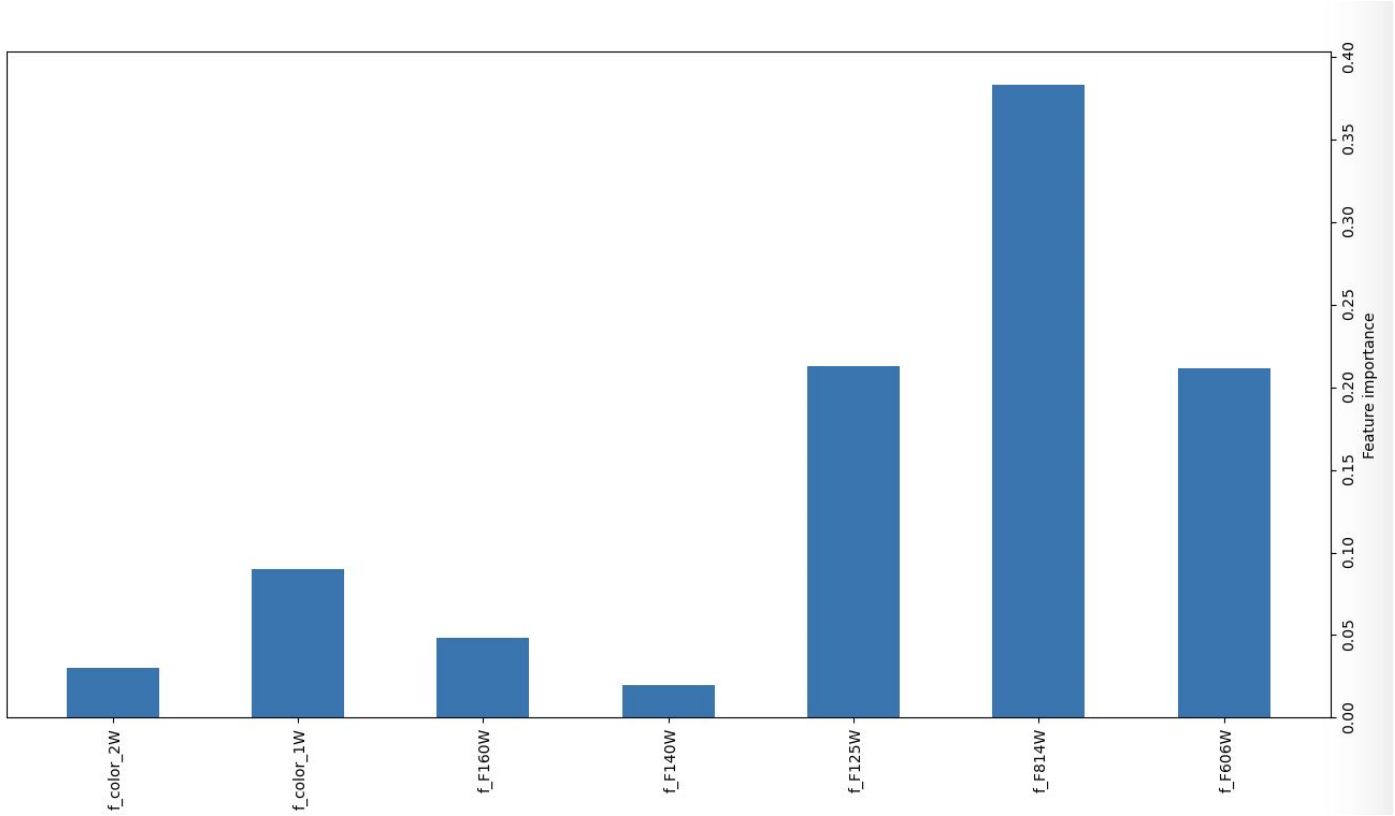
Table 1
3D-HST Fields

Field	R.A. (h m s)	Decl. (d m s)	Total Area ($arcmin^2$)	Science Area ($arcmin^2$)
AEGIS	14 18 36.00	+52 39 0.00	201	192.4
COSMOS	10 00 31.00	+02 24 0.00	199	183.9
GOODS-North	12 35 54.98	+62 11 51.3	164	157.8
GOODS-South	03 32 30.00	−27 47 19.00	177	171.0
UDS	02 17 49.00	−05 12 2.00	201	191.2

Examples of photometric points for two example galaxies



Including stellar masses above 10^9 Msun



Data preprocessing is important and could change the important features

D

A³ Net

Galaxy Cluster Masses with CNNs

Group-D : Mingyuan Zhang, Jing Dou, Xufan Hu, Yanchi Liu, Satsuki Sekiguchi, Kosei Matsumoto, Haosong Wang

Sep. 6, 2024 @Osaka University

Estimation of Galaxy Cluster Masses with CNNs

Goal: use a CNN to estimate the mass of a galaxy cluster from x-ray mock observations.

Assignments:

（1. Reproduce the paper’s results.

（2. Apply a rotationally invariant CNN,
and see if that approach improves the results!

Hot media in Galaxy clusters emit X-ray radiation primarily through bremsstrahlung
→ this emission may tell us information of the inside of galaxy clusters

Composite image of the Perseus galaxy cluster. Image credit: NASA / CXO / Fabian et al / Gendron-Marsolais et al / NRAO / AUI / NSF / SDSS.

[1] picture
https://www.sci.news/astronomy/x-ray-signal-perseus-galaxy-cluster-dark-matter-05556.html

Why machine learning?

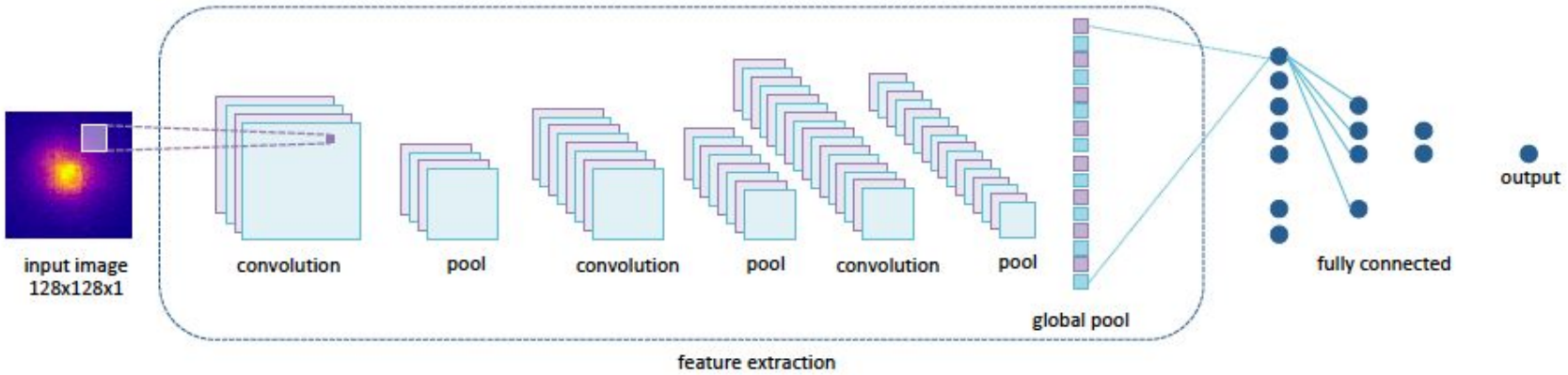
Traditional ways to estimate the cluster mass: X-ray luminosity, temperature…

Uncertainties: physical processes (e.g., gas cooling, cluster mergers) are not properly reflected in these methods… more complex…

→ **Machine learning can help us!**

Method: based on X-ray mock images of the IllustrisTNG simulation, we can develop a CNN model to accurately estimate cluster masses

Architecture of the convolutional neural network

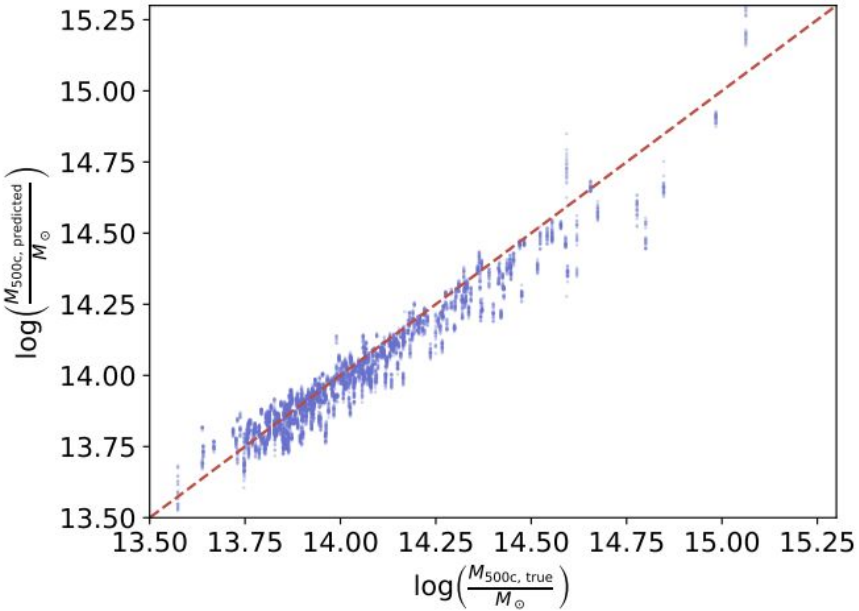


[2] M. Ntampaka, J. Zuhone, D. Einstein, et. al., 2019, ArXiv:1810.07703v2

(1. Recreation of the results

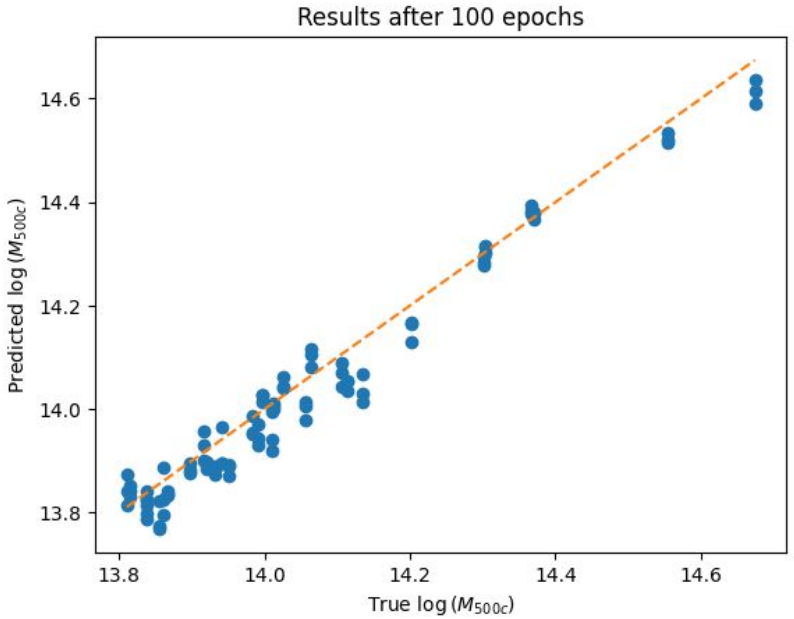
Predicted mass with CNN vs true mass

(Ntampaka,et.al.2019)



Predicted mass with CNN vs true mass

(Our work)



→ We reproduced the original result successfully.

but can we improve the accuracy for the estimation of the cluster mass?

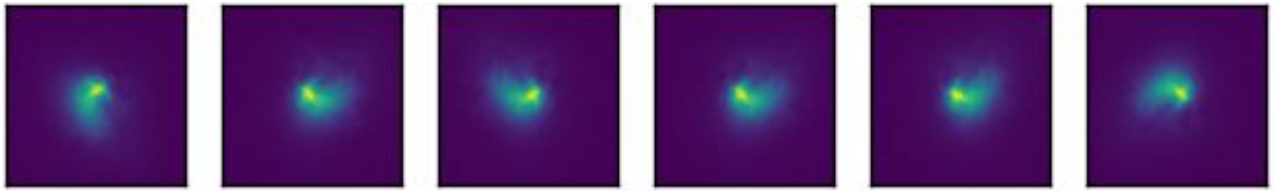
(2. Can we improve the original results?

Problems: the number of the galaxy cluster samples (986) may be too small.
there is not so many variations of the galaxy clusters.

Method: the estimation of the cluster masses is rotationally invariant to the images.
we can increase the variations and number of the galaxy clusters by rotation.

(1) Data augmentation: rotate the galaxy cluster images
(2) Rotate the filters in CNN
(3) Change the architecture of the network

Data augmentation: rotated images



Results

- Image augmentation and one more layer
- Image augmentation and new CNN
- Rotation filters

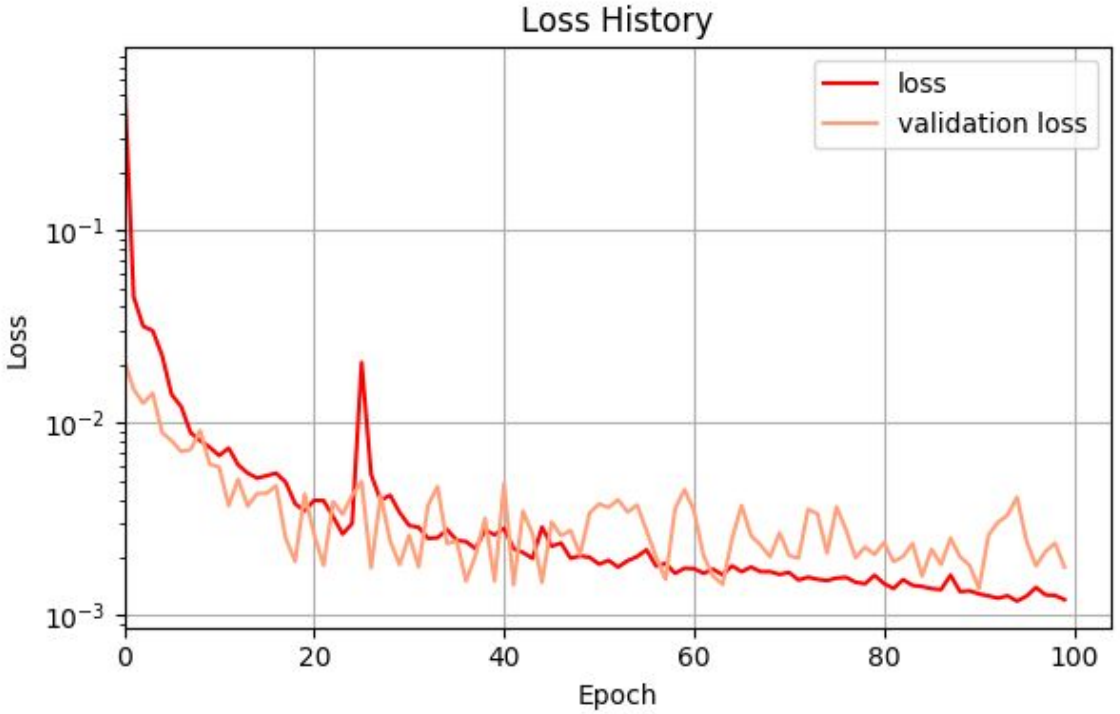
Method :Image augmentation and one more layer

- Image Augmentation
  - ImageDataGenerator(rotation_range=360, horizontal_flip=True, vertical_flip=True)
  - 10 times amount of data

- CNN structure
  - Add a new convolution layer with 128 filters

```
input_shape = (image_size, image_size, 1)
model = Sequential()
model.add(Conv2D(16, kernel_size=(3, 3), input_shape=input_shape, activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))#use AveragePooling2D

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))#use AveragePooling2D

model.add(GlobalAveragePooling2D())
model.add(Dropout(0.1))
model.add(Dense(200, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(100, activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(1, activation='linear'))
```

Training 1:Image augmentation and one more layer



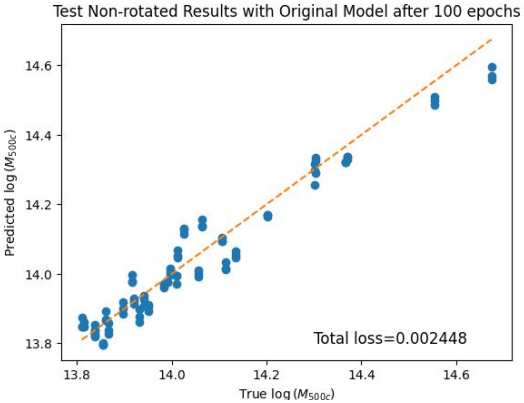
- setup
  - epochs=100
  - batchsize=16
- cost
  - platform: RTX 3090Ti
  - ~22GB
  - ~100s

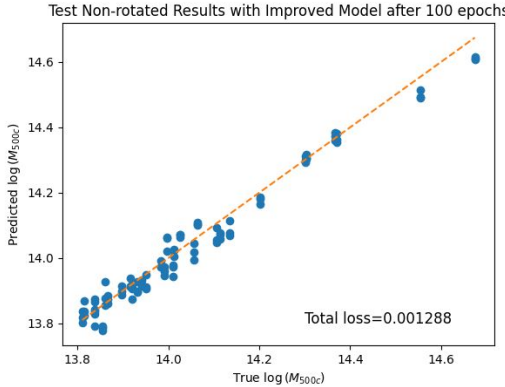
Result 1:Image augmentation and one more layer

Original Model

Improved Model

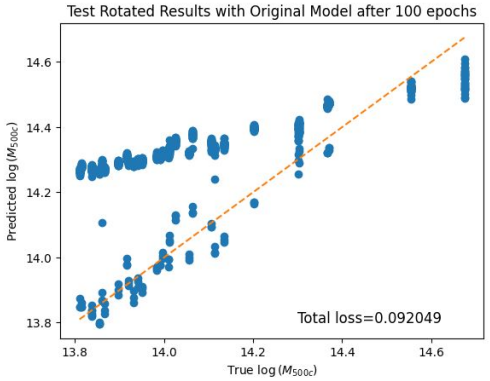
Non-rotated test set

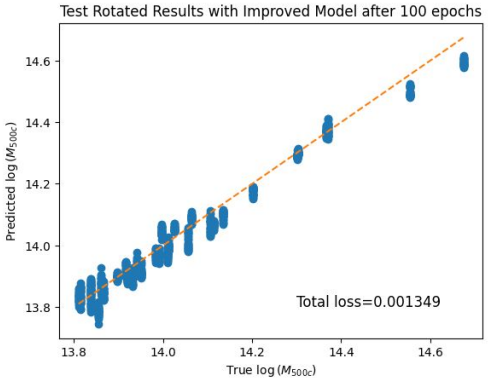




- **Non-rotated test set**
- 0.002488/0.001288~1.93

Rotated test set





- **Rotated test set**
- 0.092049/0.001349~73.70

Method 2: Image augmentation and new CNN

- **Data Augmentation**
  - ImageDataGenerator(rotation_range=**180**, horizontal_flip=True, vertical_flip=True)
  - **12** times amount of data for **train**, **validation**, and **test** dataset.

- **New CNN structure**
  - Add **more convolution layers**

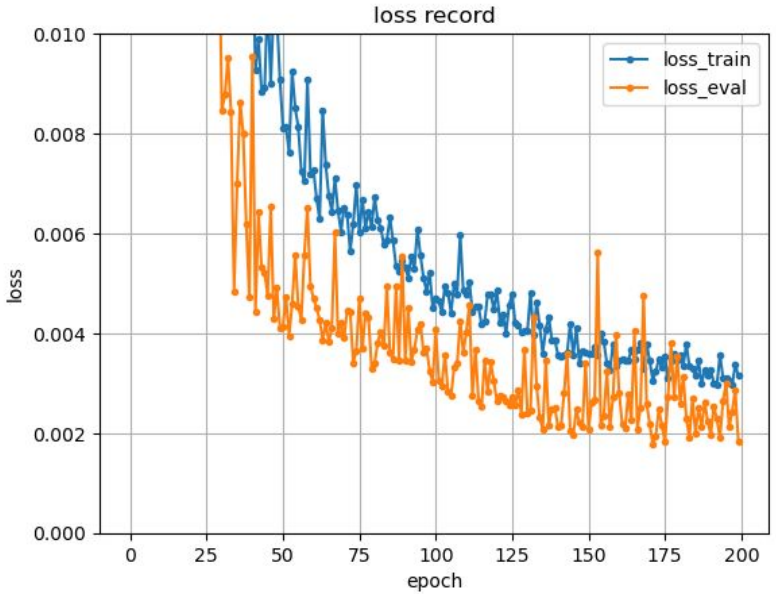
    & **Increse filters** of convolution layers (16,32,64 ⇒ 32,32,64,64,128,128)

  - Use **AveragePooling** to replace MaxPooling at the end
  - **Change filters** of Dense layers

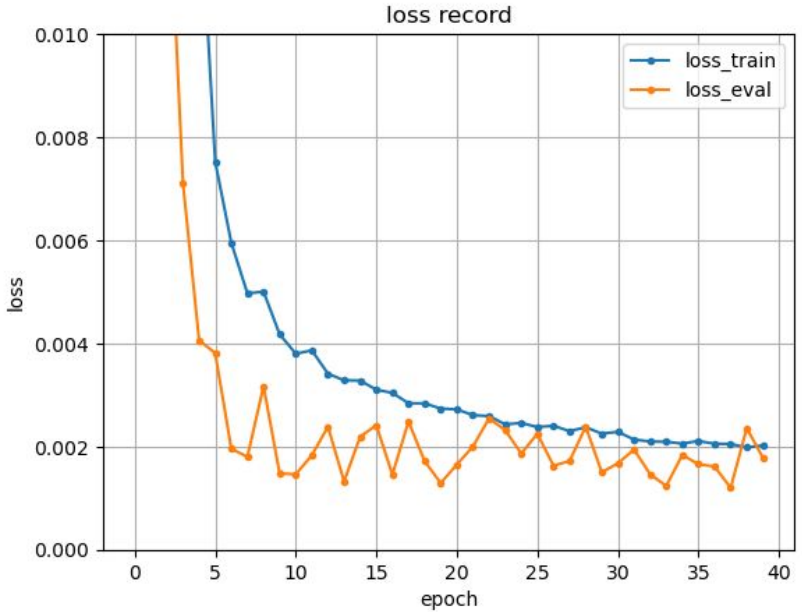
```
input_shape = (image_size, image_size, 1)
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), strides=1, padding="valid", input_shape=input_shape, activation='relu'))
model.add(Conv2D(32, kernel_size=(3, 3), strides=1, padding="valid", activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(64, kernel_size=(3, 3), strides=1, padding="valid", activation='relu'))
model.add(Conv2D(64, kernel_size=(3, 3), strides=1, padding="valid", activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(128, kernel_size=(3, 3), strides=1, padding="valid", activation='relu'))
model.add(Conv2D(128, kernel_size=(3, 3), strides=1, padding="valid", activation='relu'))
#28*28*128
model.add(AveragePooling2D pool_size=(4, 4), strides=4)) # 使用平均池化代替最大池化，可以追踪tail特征
#7*7*128
model.add(GlobalAveragePooling2D())
#128
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='linear'))
```

The structure of improved model

Training 2: Image augmentation and new CNN



Loss of original model (epochs = 200)



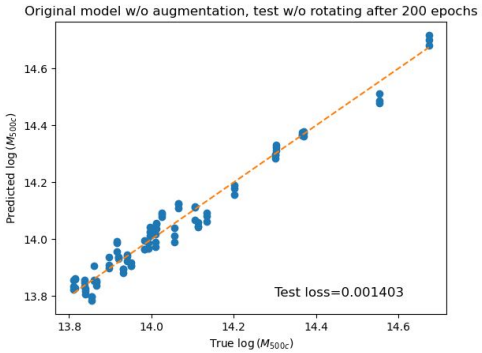
Loss of improved model (epochs = 40)

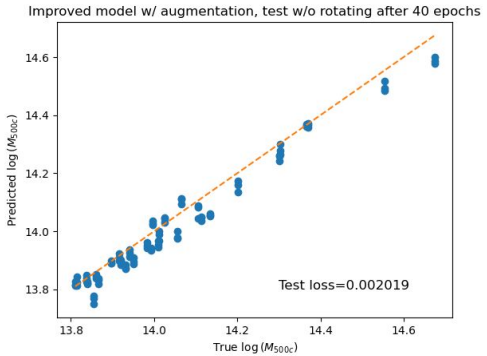
Result 2: Image augmentation and new CNN

Original Model

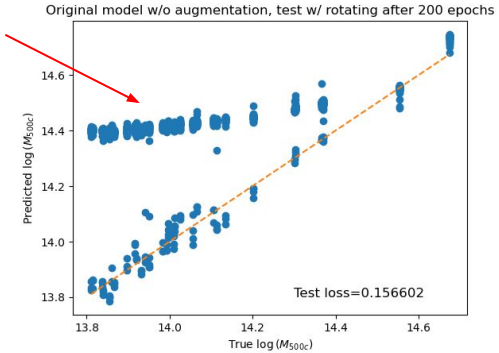
Improved Model

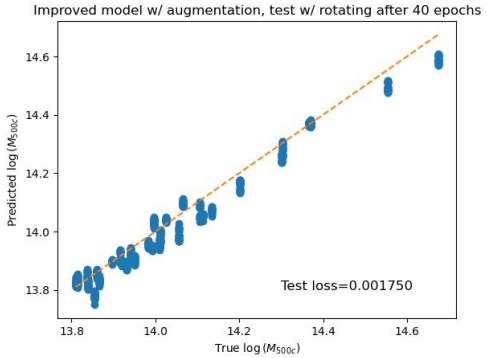
Non-rotated test dataset





Rotated test dataset





- **Similar loss** of **both models**, evaluated through **non-rotated test dataset**
- **Original model fails** to distinguish the rotated images ❌
- **Improved model can!** ✅

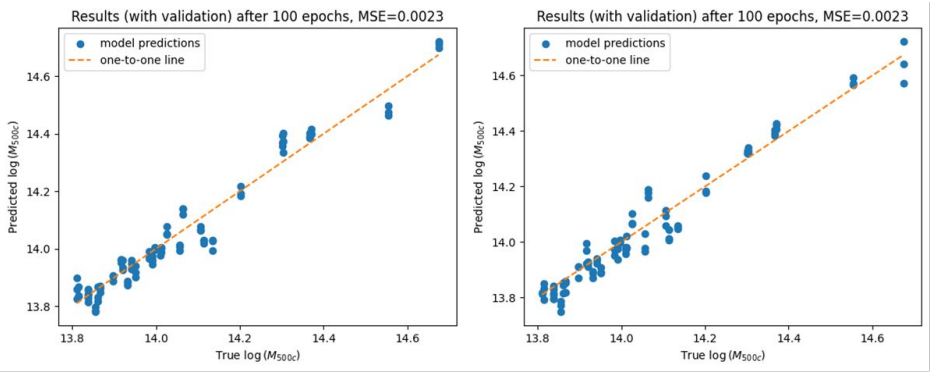
Method 3: Model augmentation with rotation filters

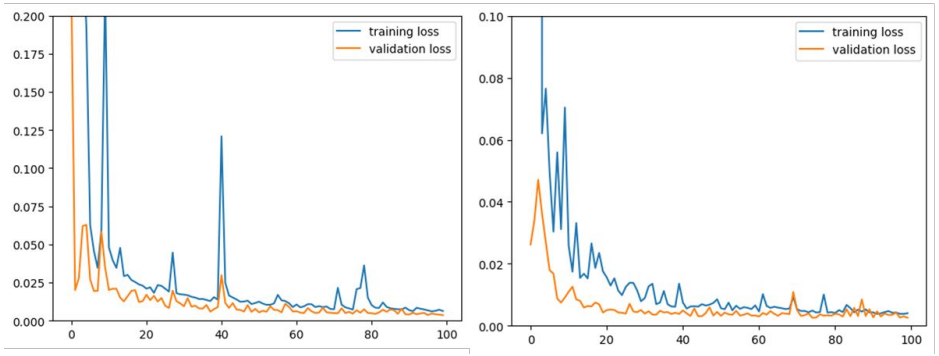
- With the class of keras.layers.RandomRotation, we add an extra rotational layer into the CNN, attempting to achieve a Rotational invariant CNN.

- The RandomRotation class contains 5 parameters, here are 2 important ones:
  - factor: it controls the rotation degree of the filter, if factor = 0.2, we are going to rotate the filter in a random degree between (-0.2*2pi, 0.2*2pi).
  - fill_mode: it controls how we fill the missing pixels, 'nearest' is usually a good choice.

Result 3: Model augmentation with rotation filters

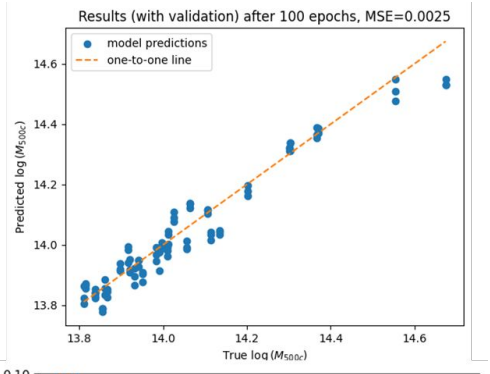
1 rotational layer, factor = 0.01

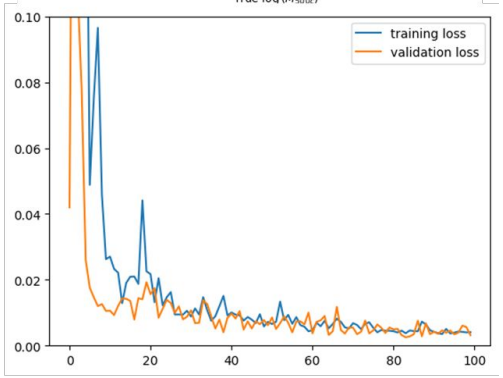




1 rotational layer, factor = 0.1

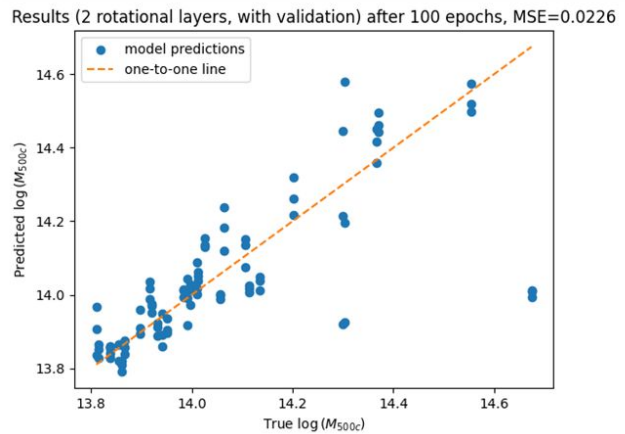
1 rotational layer, factor = 0.2



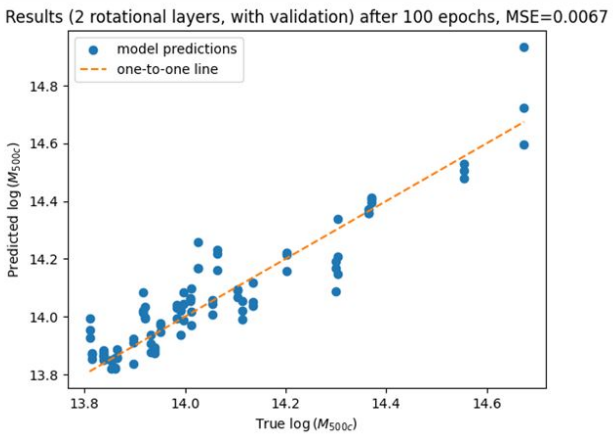


Result 3: Model augmentation with rotation filters

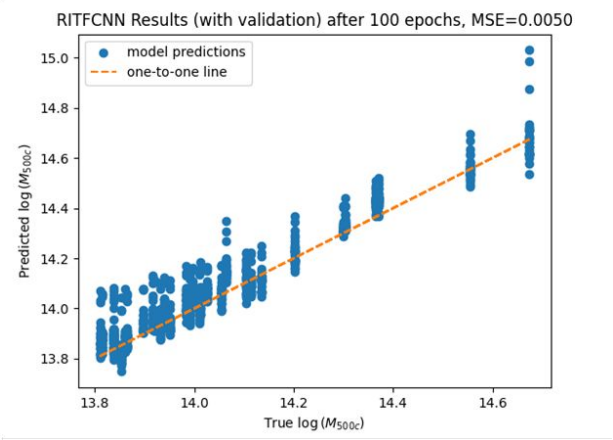
2 rotational layers, factor=0.1



3 rotational layers, factor=0.1



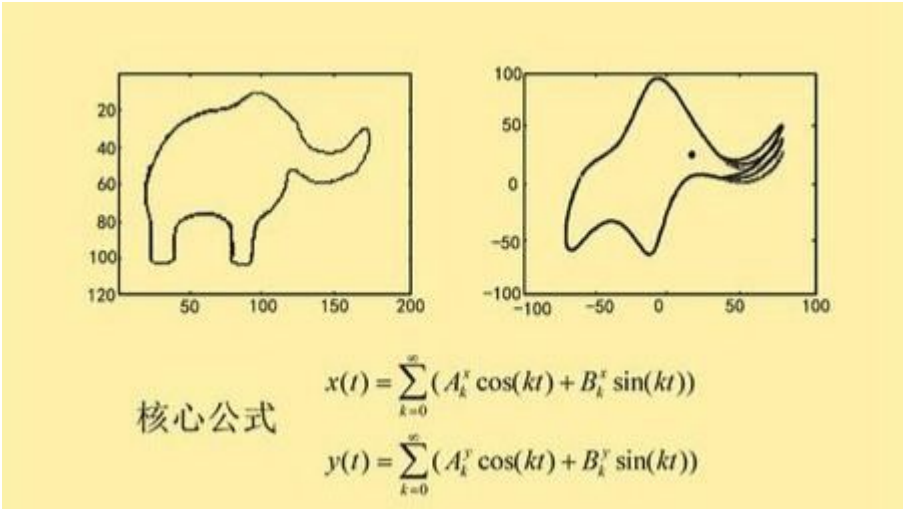
1 rotational layer,
1 transition layer,
1 flip layer
Max Ultra Plus version



Run 10 training and the result is not good at all.

Discussion

(1) Interpret result?
(2) other CNNs
(3) risks of more layers



Conclusion

- We tried to improve the estimation the cluster mass from X-ray mock images with **ROTATION**.
- The smaller the rotation degrees are, the better result we get, but not that good.
- Extra layers of reflection and SO2 symmetry are not helpful, for they may mess up the image size and pixels thus resulting in more problems.

Methods	the ratio “original loss / our loss”
data augmentation	not improved
data augmentation + adding Conv. layer	1.93
data augmentation + adding Conv. layer + increasing filters	1.44
model augmentation with rotation filters	about 1.50, but not stable

Reference：

[1] https://www.sci.news/astronomy/x-ray-signal-perseus-galaxy-cluster-dark-matter-05556.html
[2] M. Ntampaka, J. Zuhone, D. Einstein, et al., (2019), arXiv:1810.07703v2

E

Using CNNs to infer the fundamental parameters of the Universe

Presenter: Zhao ZHANG, Tao JI, Nobuyuki SAKAI

Code: Zhao ZHANG (CNN), Zitao HU, Xiaoyue CAO(ResNet18), Tao JI(Keras)

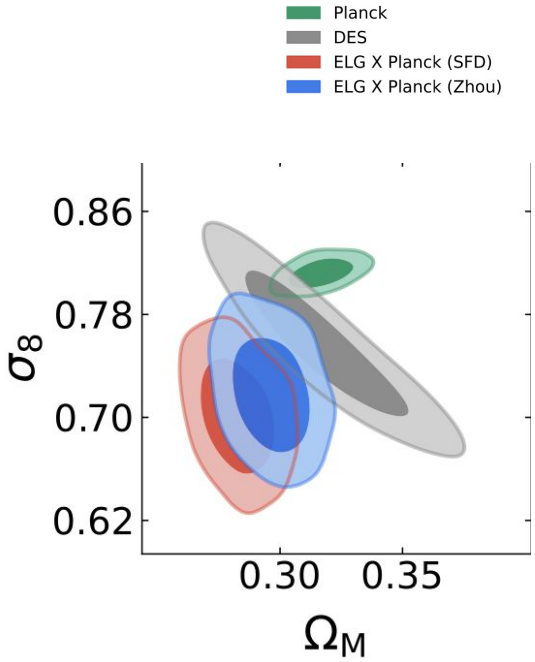
PPT slides: Tao JI, Zhao ZHANG, Yushan XIE, Xiaoyue CAO

Discuss: Nobuyuki SAKAI, Seo-eun LEE

Thanks: Leander Thiele & Michelle Ntampaka

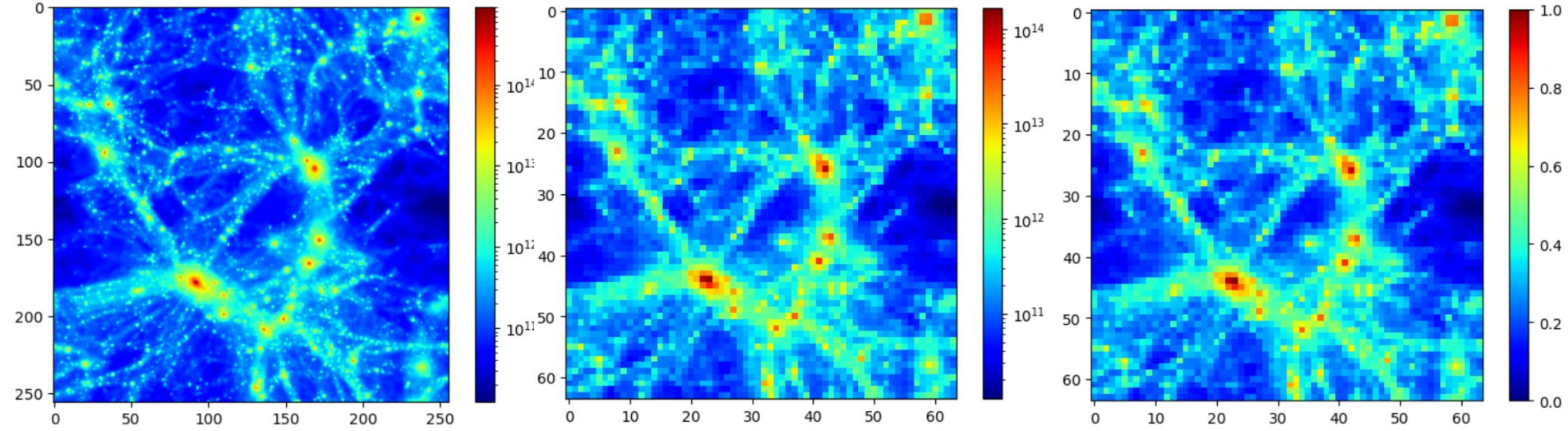
Why interesting?

- Challenge: precise estimation of cosmological parameters validating cosmological models and unlocking the puzzles of the Universe such as Hubble and S8 tensions
- Inferring cosmological parameters from highly non-linear structures is challenging for classical methods based on explicit physical models.
- Constrain the feedback process of the Universe like Active galactic nucleus (AGN) and Stellar feedback (SN, SN wind).
- For current simulation, we unable to return the starting point (nonlinear disturbance)
- For us, CNN is new thing.



Karim+24

Data preprocessing



Original data
(256✖256)

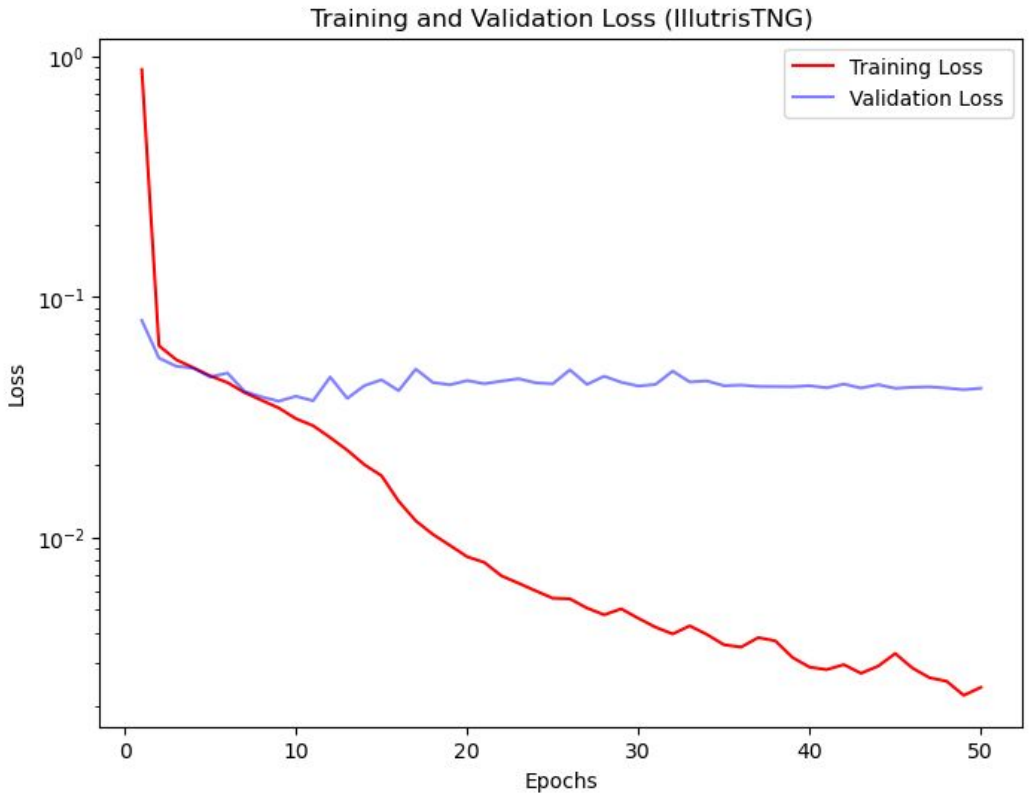
Downsampled data
(64✖64)

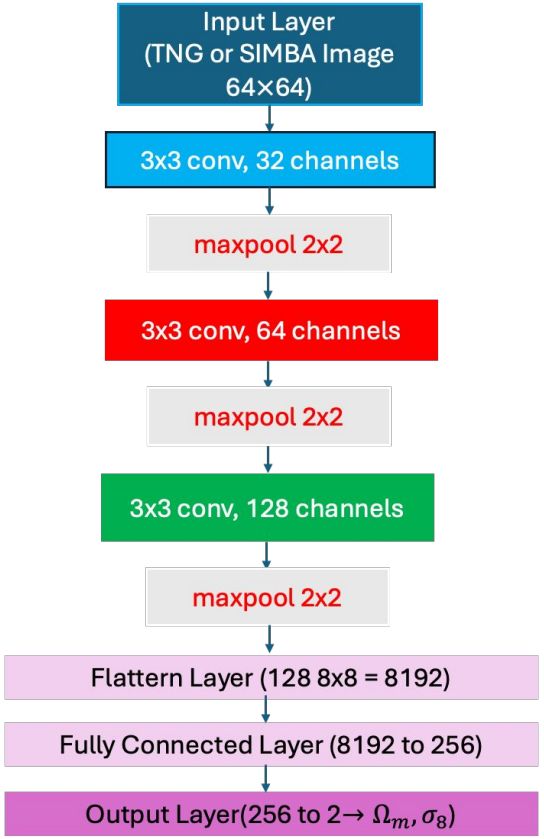
Logarithm and Normalization

We have 1000 (simulations)✖15 (field of view) pictures
with 2 observables (Mtot and P,
and 6 parameters Ωm, σ8, SN1, AGN1, SN2, AGN2)
in IllustrisTNG or SIMBA simulation

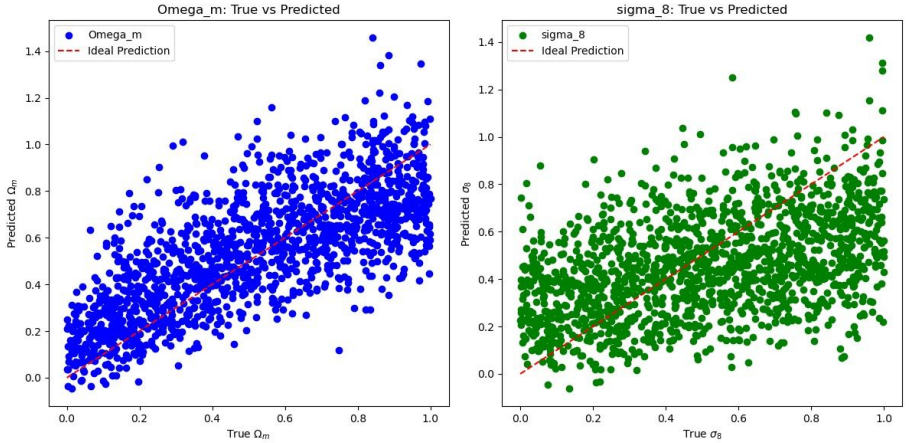
Traditional CNN Model Architecture (Pytorch)

train and test using IllustrisTNG data (7:2:1 train:validation:test)



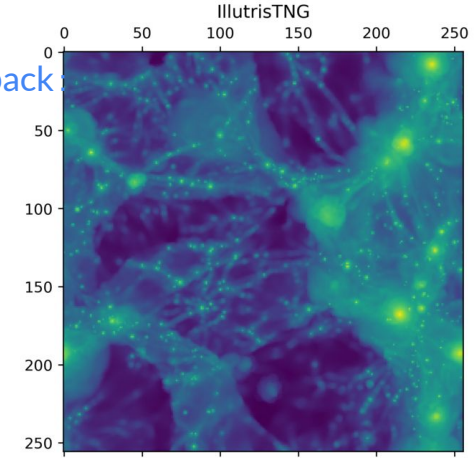


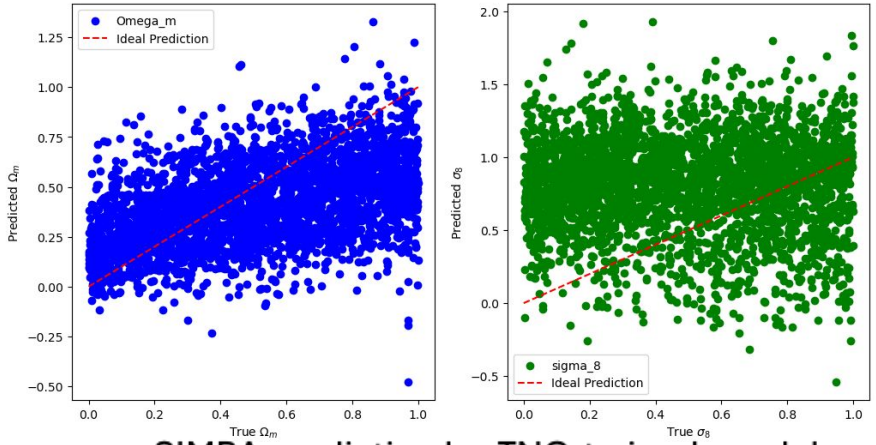
Train using IllustrisTNG data (7:2:1 train:validation:test) and test on both



IllustrisTNG train set

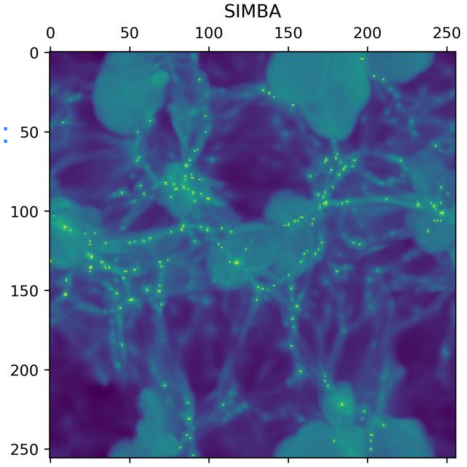
Soft feedback:





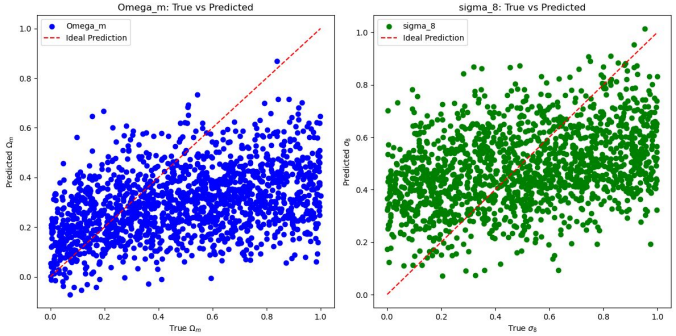
SIMBA prediction by TNG-trained model

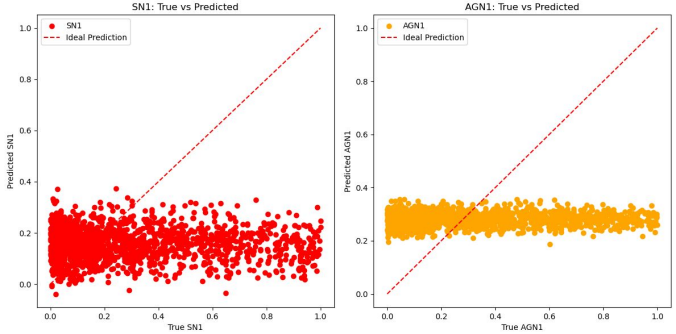
Strong feedback:

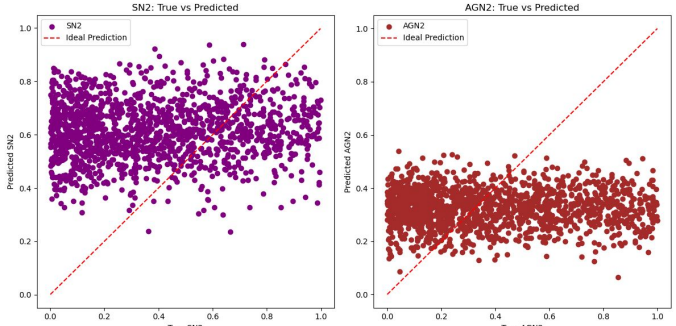


6 parameters comparison between IllustrisTNG and SIMBA

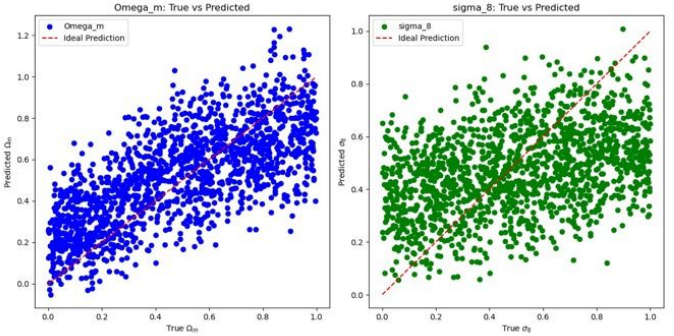
Illustris TNG:

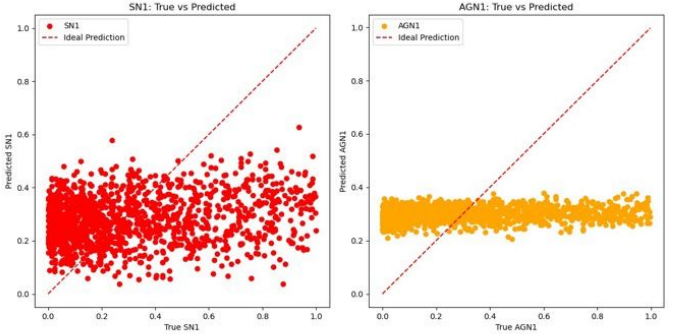


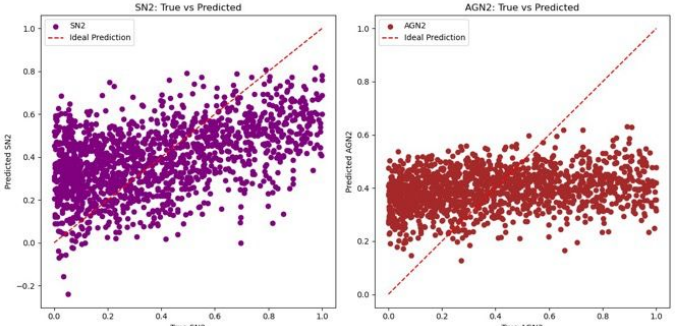




SIMBA:

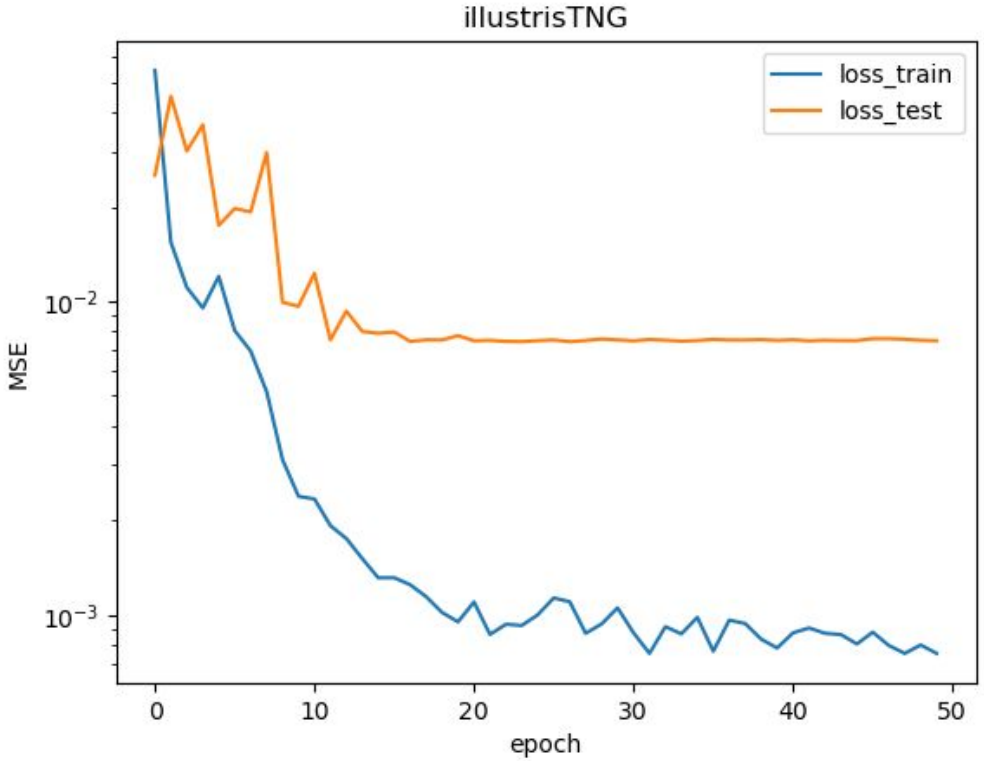






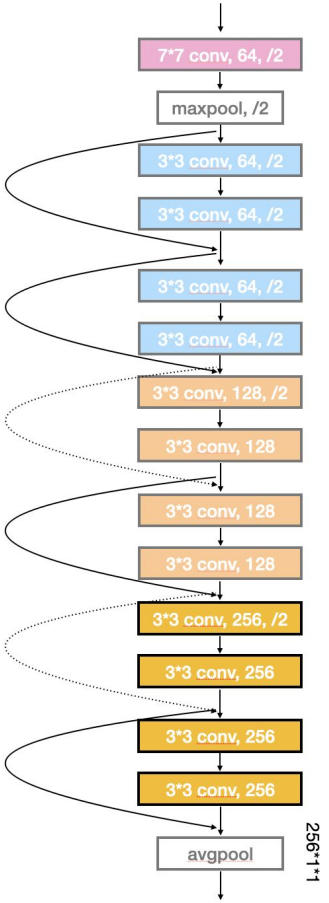
ResNet18 Model Architecture (Pytorch)

train and test using IllustrisTNG data (6:2:2 train:validation:test)

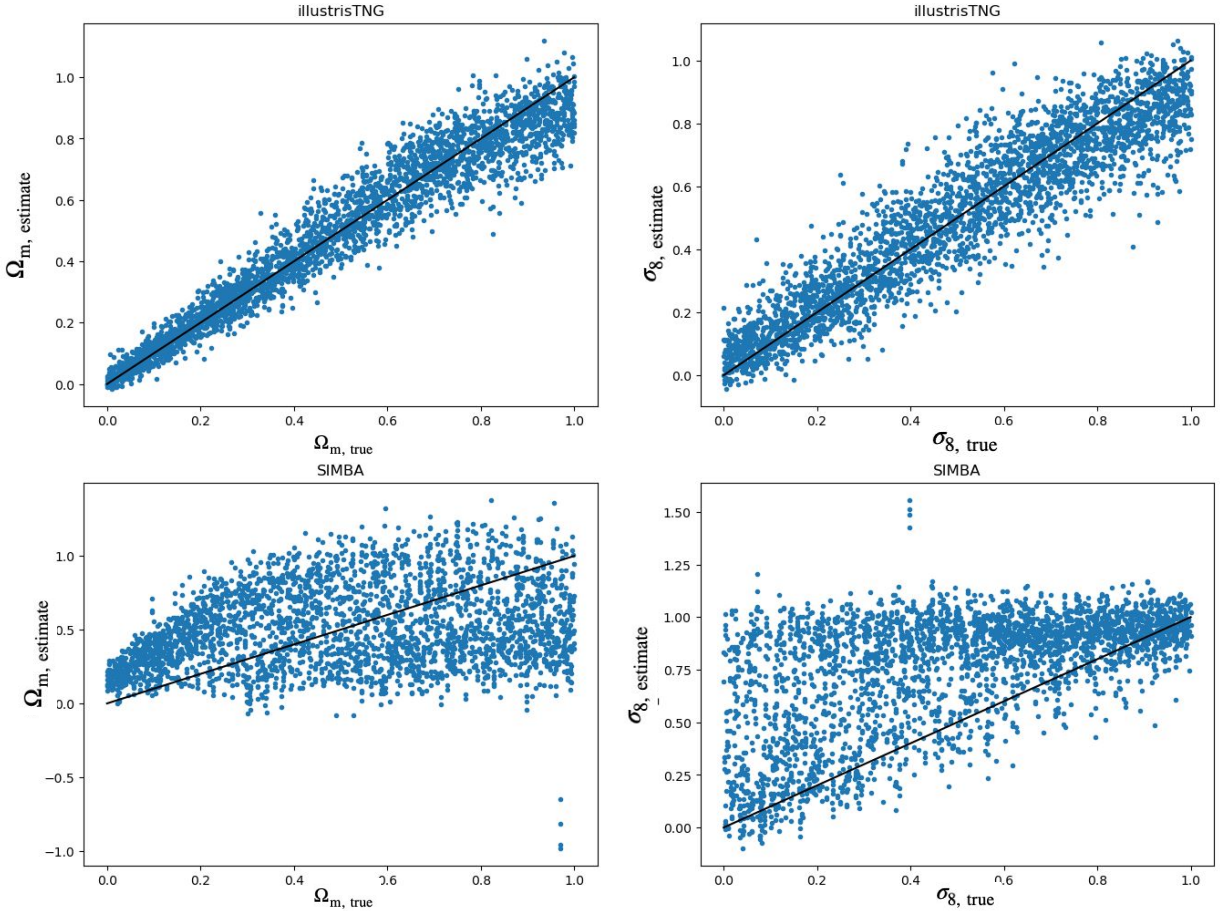


Classical ResNet18 structure

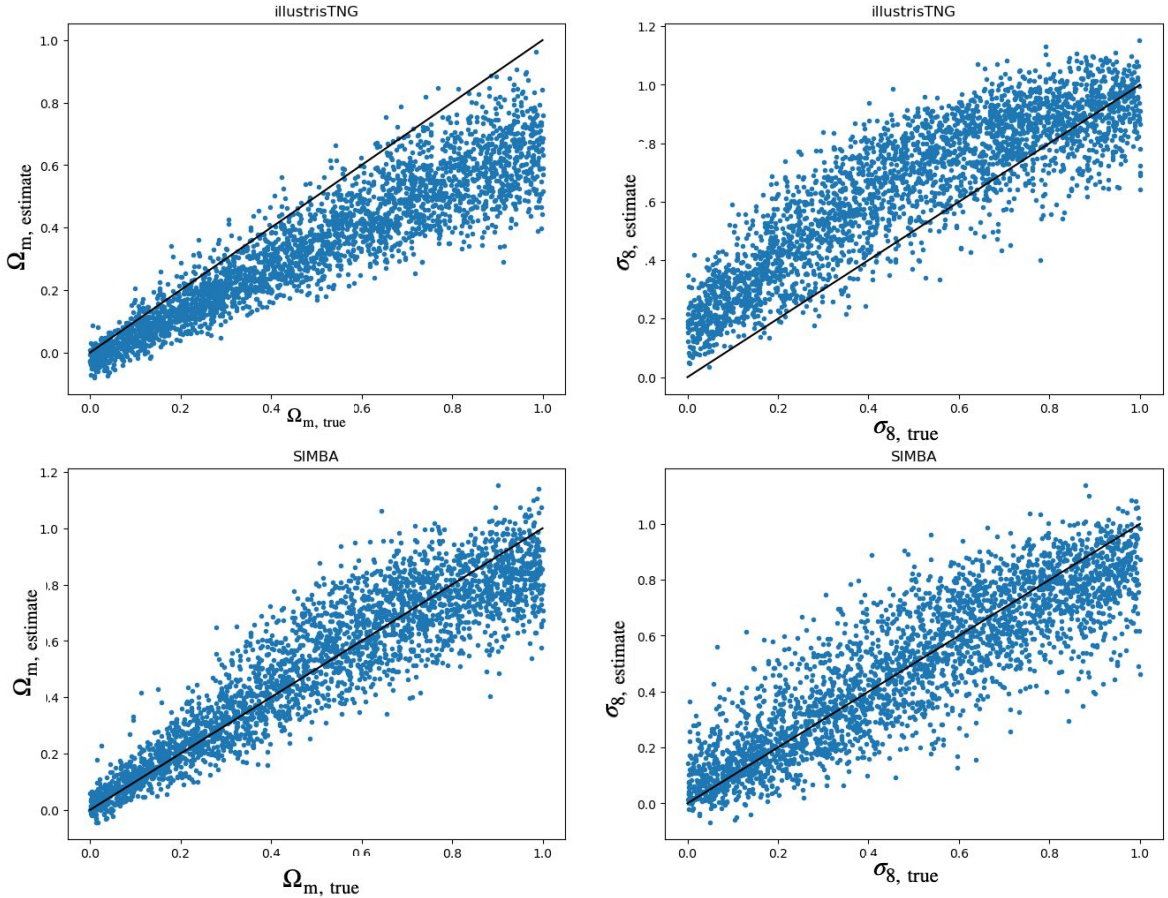
We remove three layers to reduce computational overhead



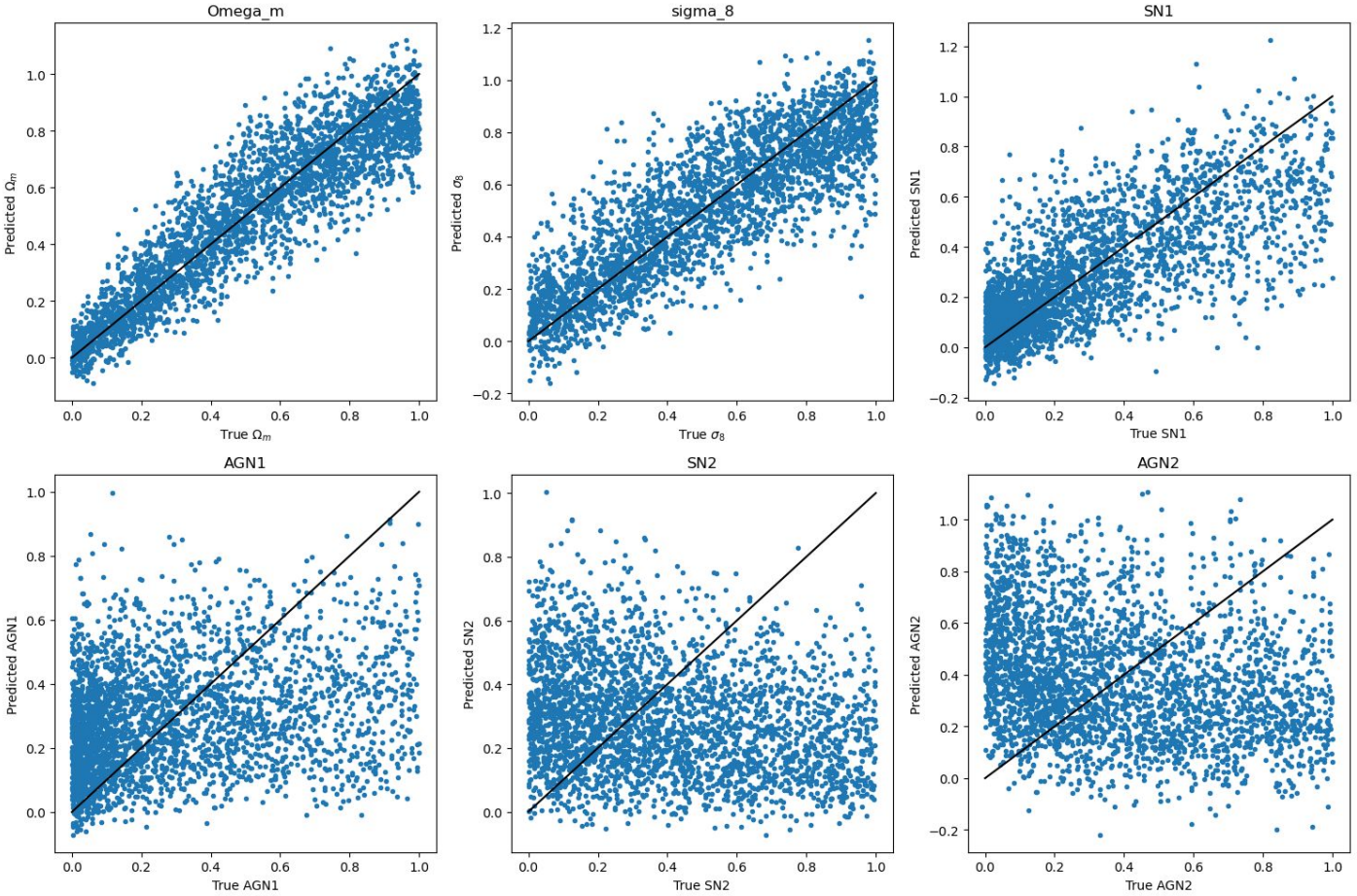
ResNet18
train using IllustrisTNG data (6:2:2 train:validation:test) and test on both



ResNet18
train using SIMBA data (6:2:2 train:validation:test) and test on both



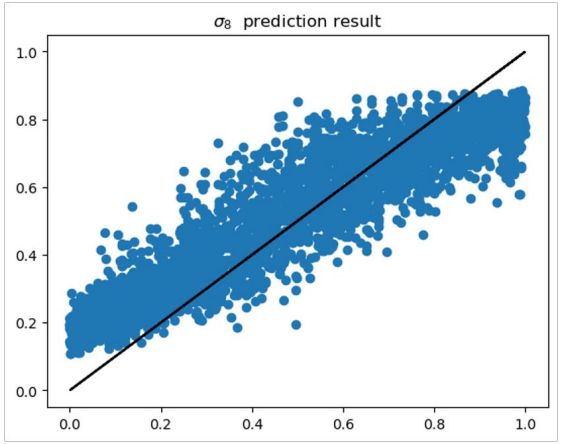
Constraining all 6 parameters of SIMBA:

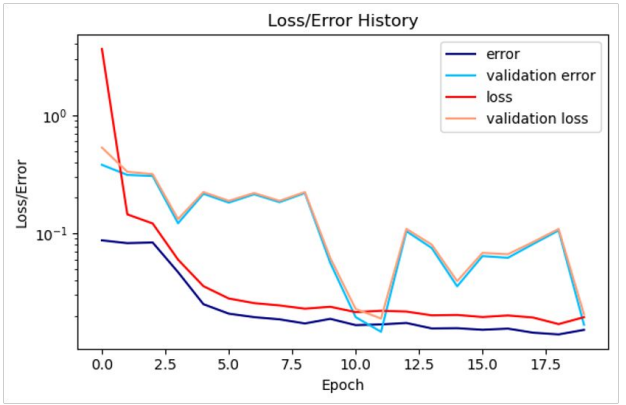


Keras

train and test on IllustrisTNG (7:3:1 train:validation:test)

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 64, 64, 2)	0
conv2d (Conv2D)	(None, 64, 64, 8)	408
batch_normalization (BatchNormalization)	(None, 64, 64, 8)	32
max_pooling2d (MaxPooling2D)	(None, 32, 32, 8)	0
dropout (Dropout)	(None, 32, 32, 8)	0
conv2d_1 (Conv2D)	(None, 32, 32, 32)	2,336
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_2 (Dropout)	(None, 8, 8, 64)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 1024)	4,195,328
dense_1 (Dense)	(None, 32)	32,800
dense_2 (Dense)	(None, 1)	33





Discussion

Data reprocessing: downsample and normalization

Model building: Hyperparameter setting + layer number

Labels of IllustrisTNG: Ω_m, $\sigma 8$; Labels of SIMBA: Ω_m, $\sigma 8$,SN1

The more, the better?

Evaluate the results

How to understand?

THANK YOU

F

A³ Net

Group F Presents: Illuminating the Future of AI with Image Mastery

Exploring Image Classification, Regression, and Generation with AI

Starring:

Hongxuan
Imkwang
Shijiao
Temurbek
Chen
Kenji

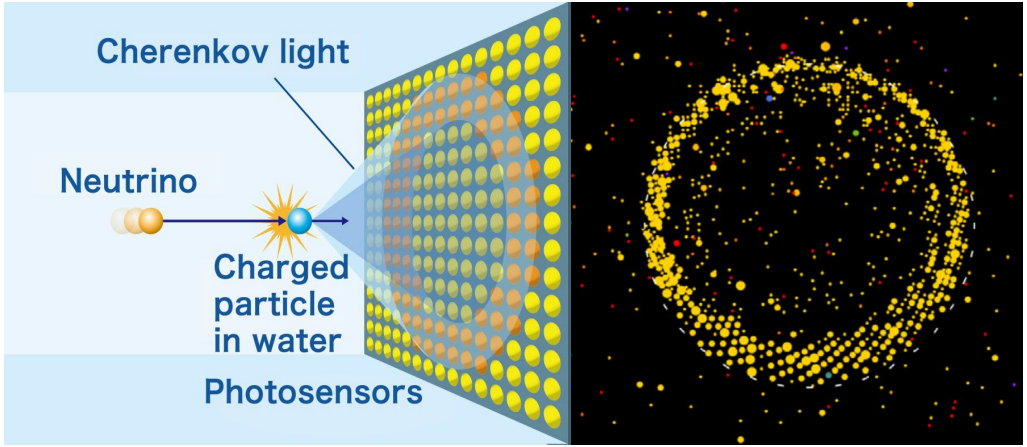
Astro AI Asian Network Summer School 2024, September 6, 2024

Our Challenge

Neutrino Telescopes:
Neutrino telescopes have been crucial in forming our current understanding of astrophysics.

Cherenkov Light:
To detect neutrino phenomena, Cherenkov light is ideal due to its distinct and predictable patterns.

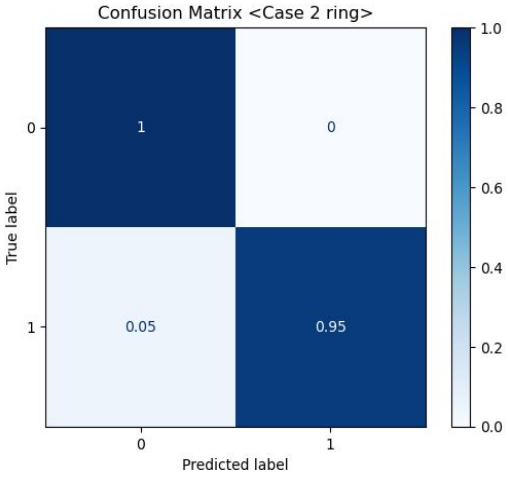
Our Mission:
Analyze Cherenkov light from neutrino detectors using AI techniques, focusing on
(i) classification,
(ii) image generation,
(iii) and regression.

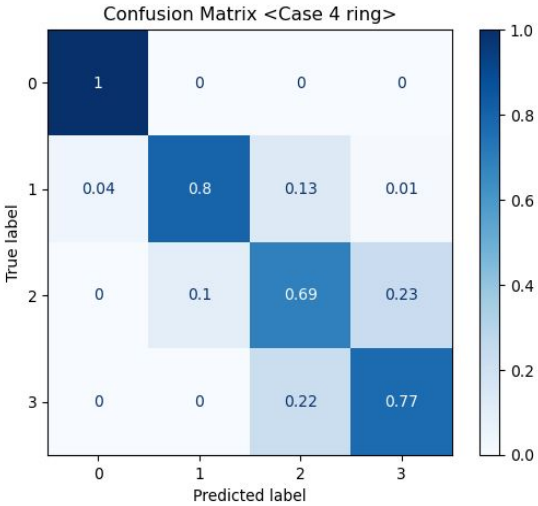


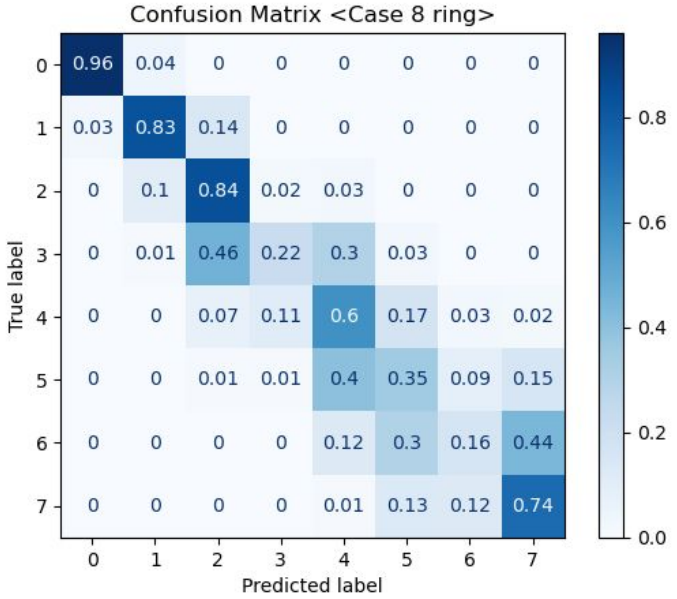
NRings Classifier

10000 of (28 pixels)^2 images
CNN: 3 convolution layers
+2 Fully connected layers
Data : Training: 7000 / Testing: 3000
Training Epoch: 5

Q: How do the performances change versus max number?



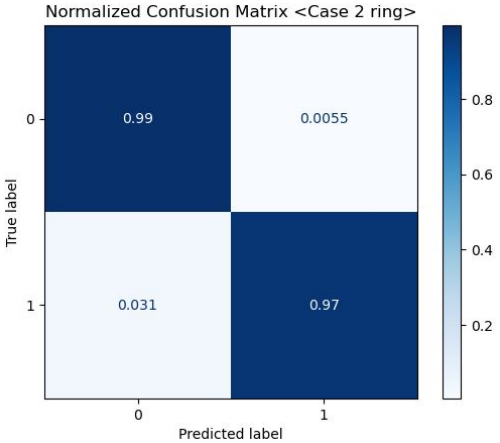


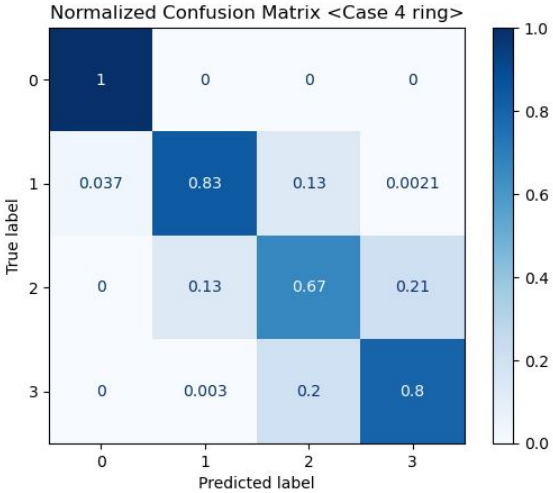


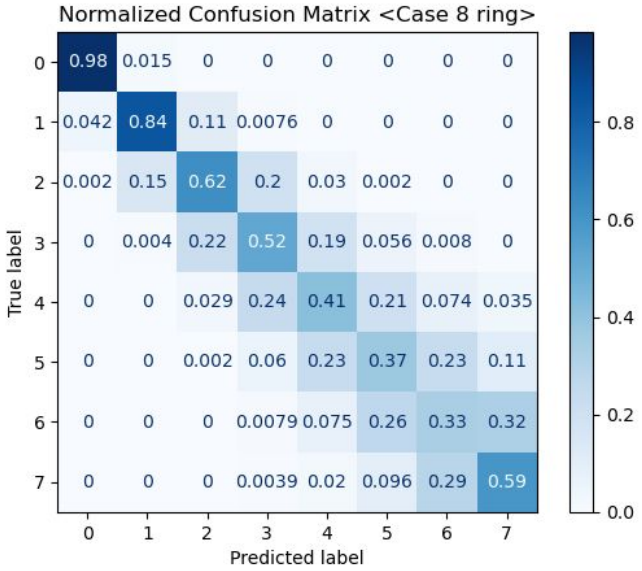
Q: Can you use a non Neural-Network-based technique to do this classification?

Principal Component Analysis (PCA): Extract important latent vectors from all images. We used top 50 principal components.

Support Vector Machine (SVM): Separate the data into different numbers of rings using a 49-dimensional hyperplane.





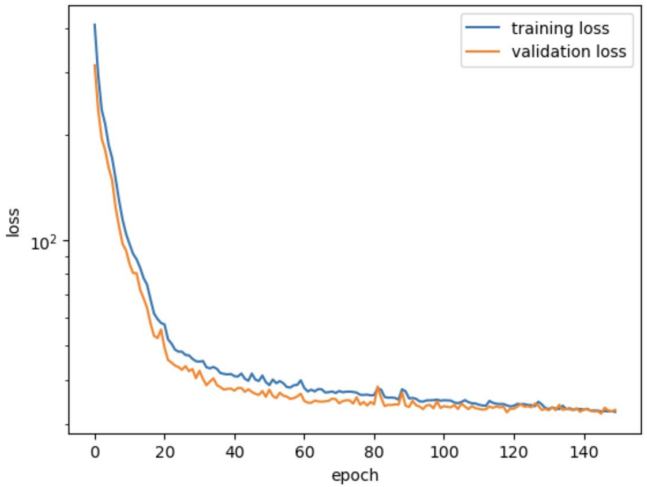


Generating simulated images

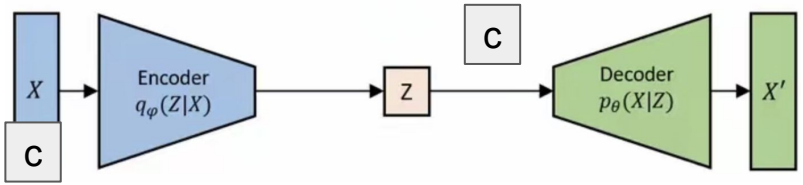
-> Use conditional-VAE model

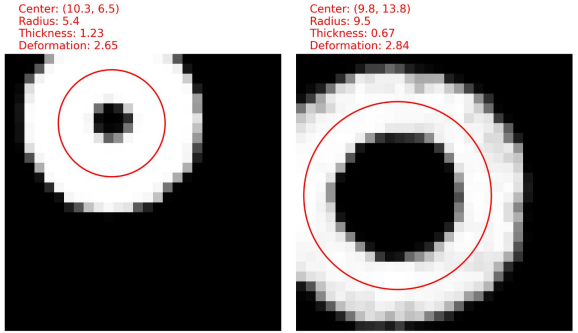
-> Fiducial: Training: 4000; Validation: 1000

-> Training epoch ~ 150



Conditional Variational Autoencoder



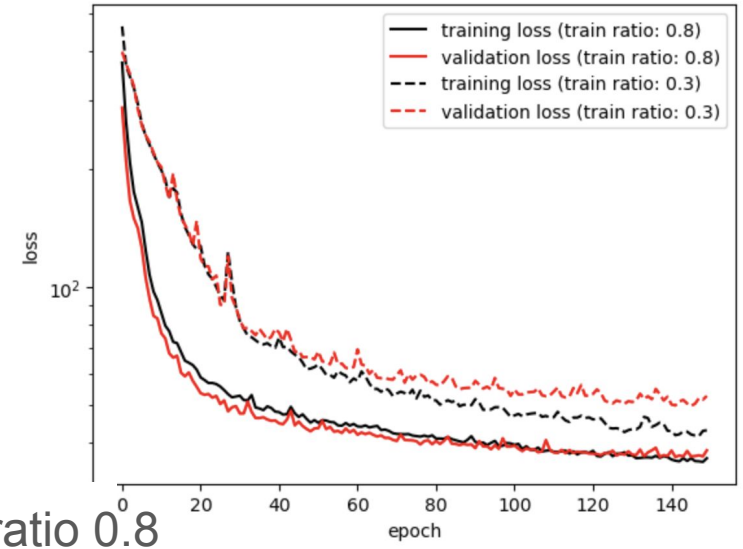


Generating simulated images

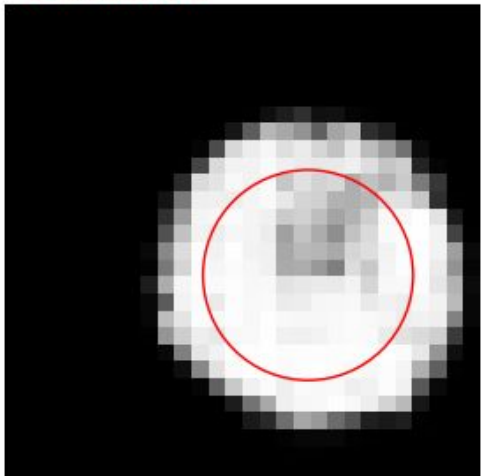
Q: How do the performances change with training set?

-> Use conditional-VAE model

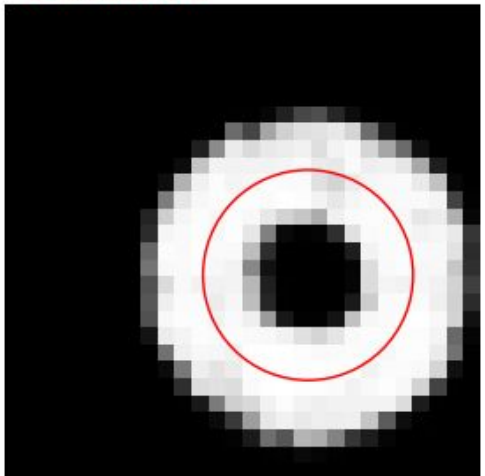
-> Different ratios between training and validation samples make different loss rates.

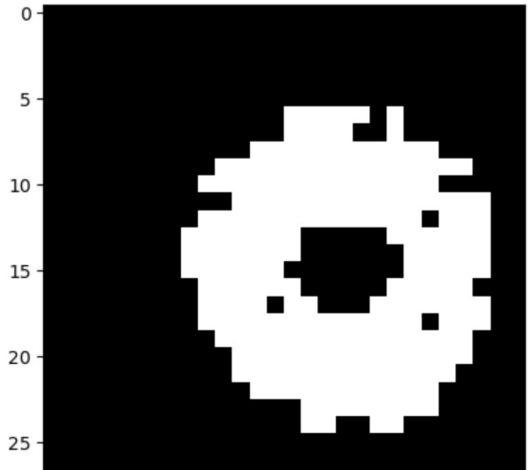


train ratio 0.3

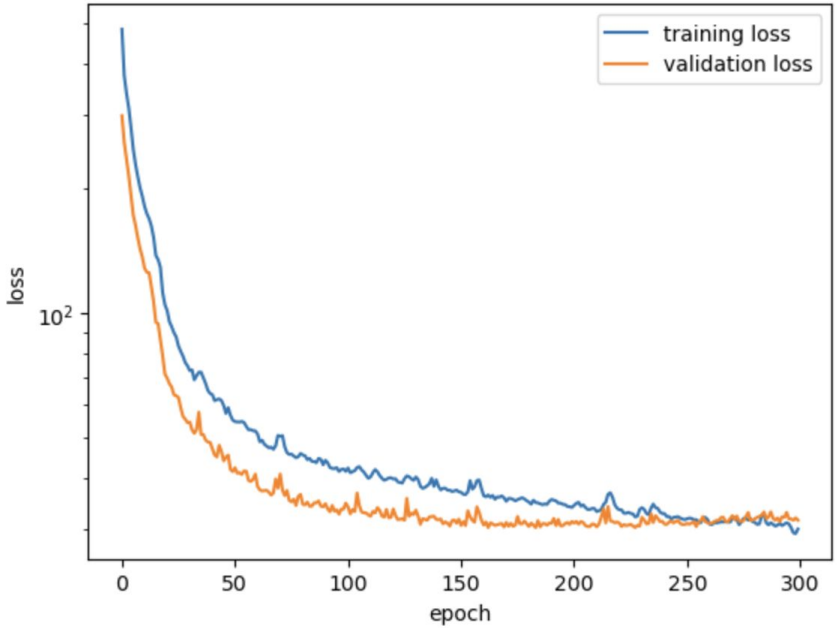


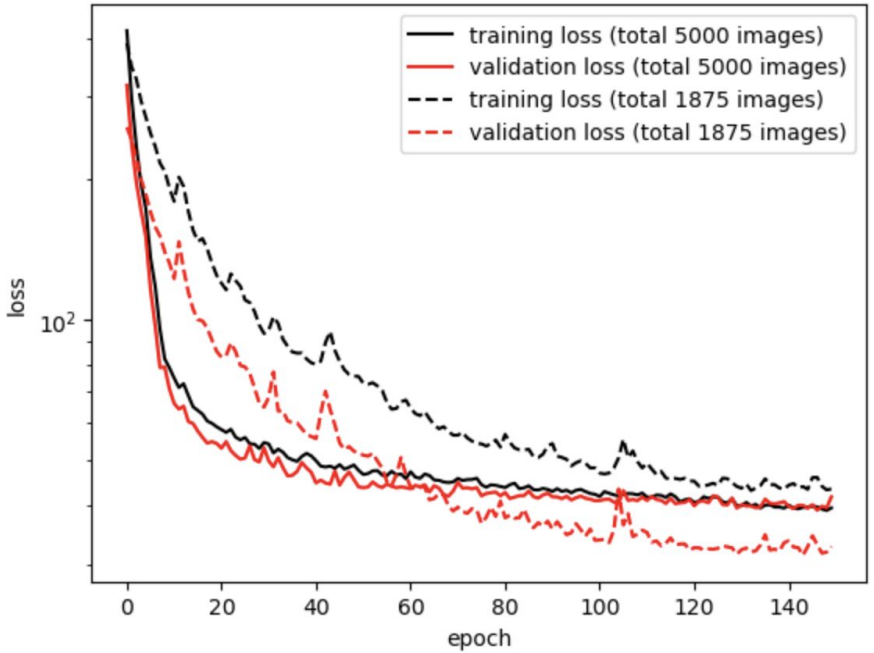
train ratio 0.8





Reduce total number of the dataset with the same training ratio





Ring Regressor

Q. Can you write some algorithm to perform regression?

CNN: 2 convolution layers+2 Fully connected layers

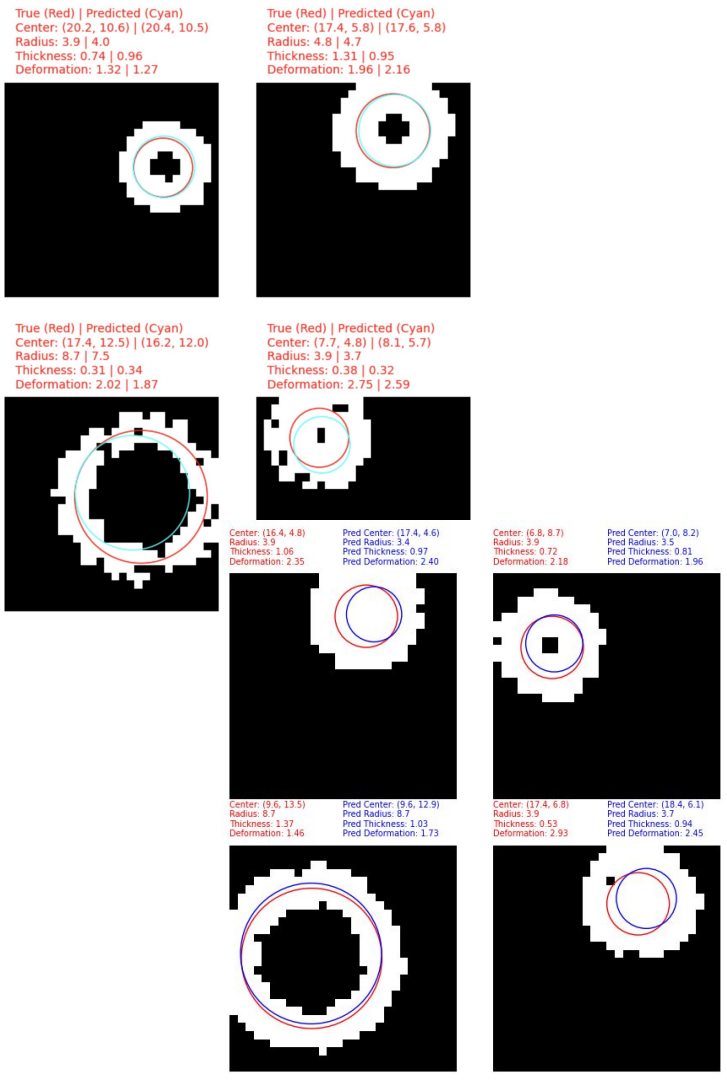
Data: Training: 8000 / Testing: 2000

Training Epoch: 8

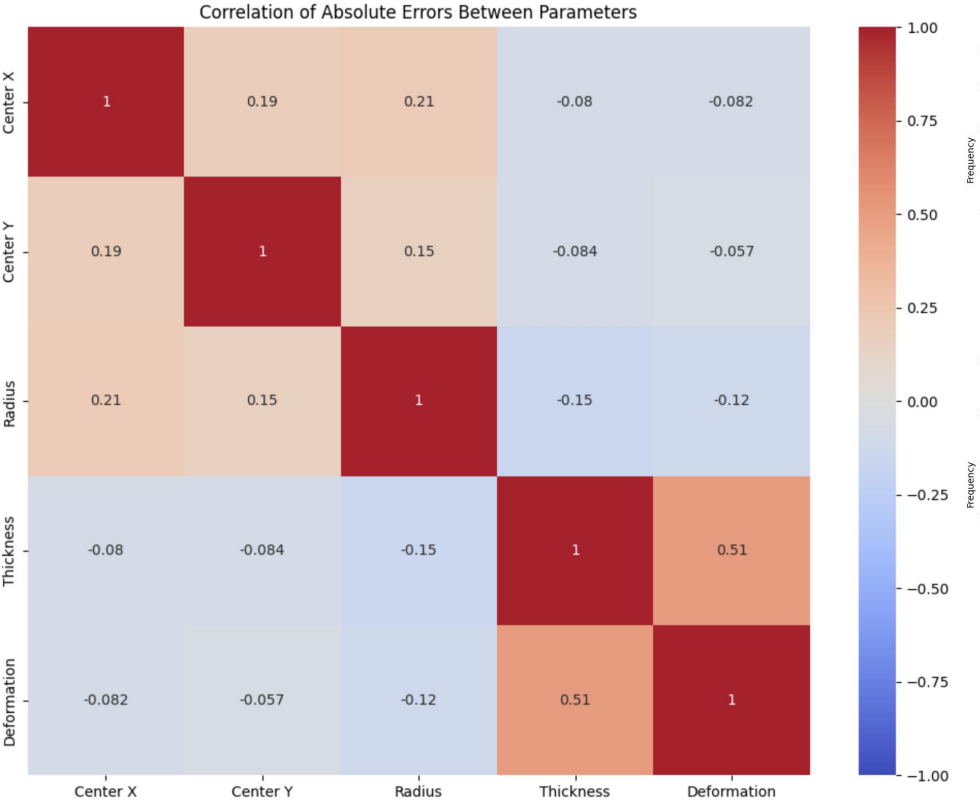
Fully Connected Network: 4 Fully connected layers

Data : Training: 8000 / Testing: 2000

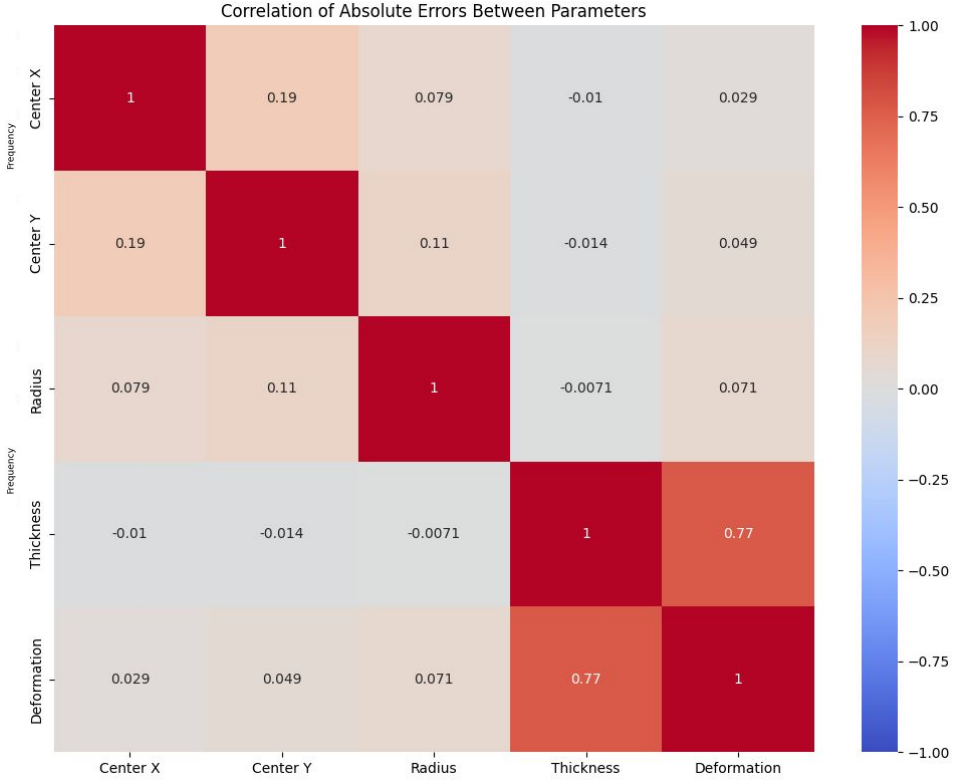
Training Epoch: 8



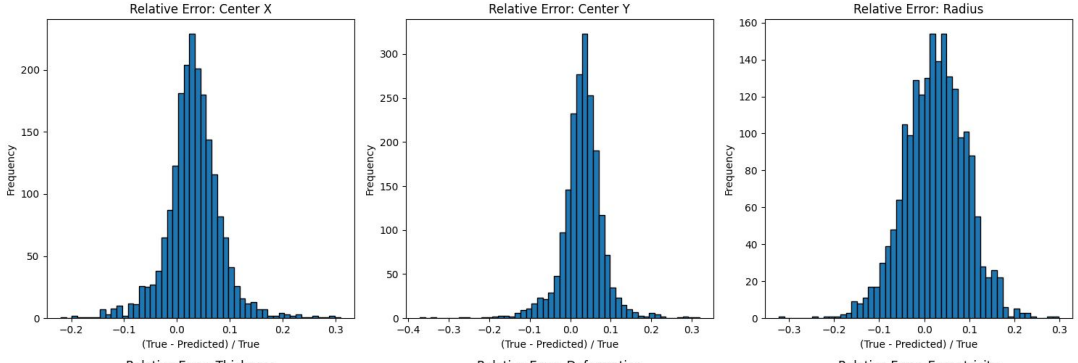
CNN

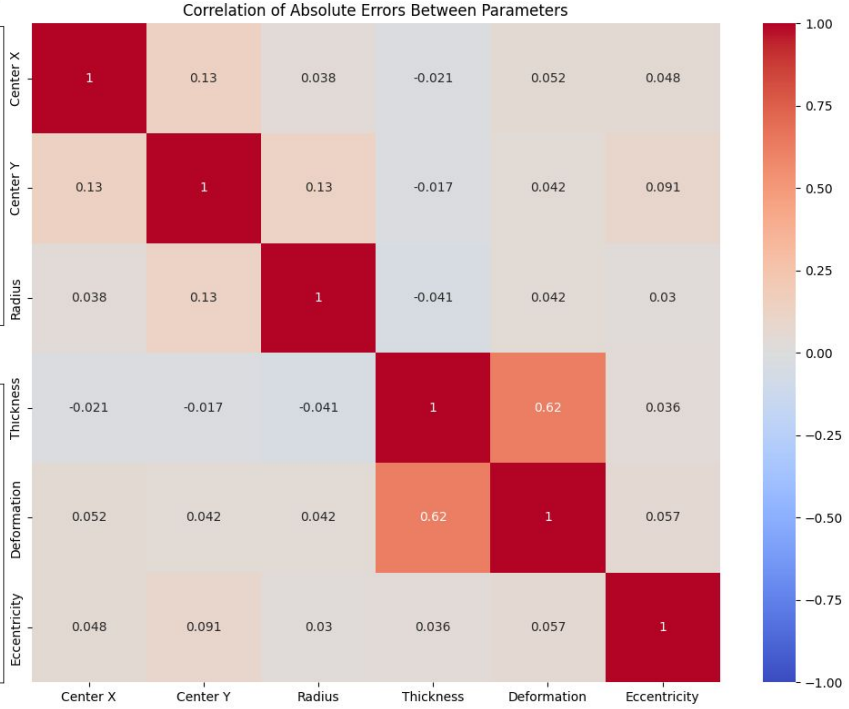


Fully Connected



Ellipse Regressor






```
└──Linear: 2-7                          774
==========================================================================
Total params: 566,918
Trainable params: 566,918
Non-trainable params: 0
==========================================================================
```

Conclusion and Applications

AI Success:

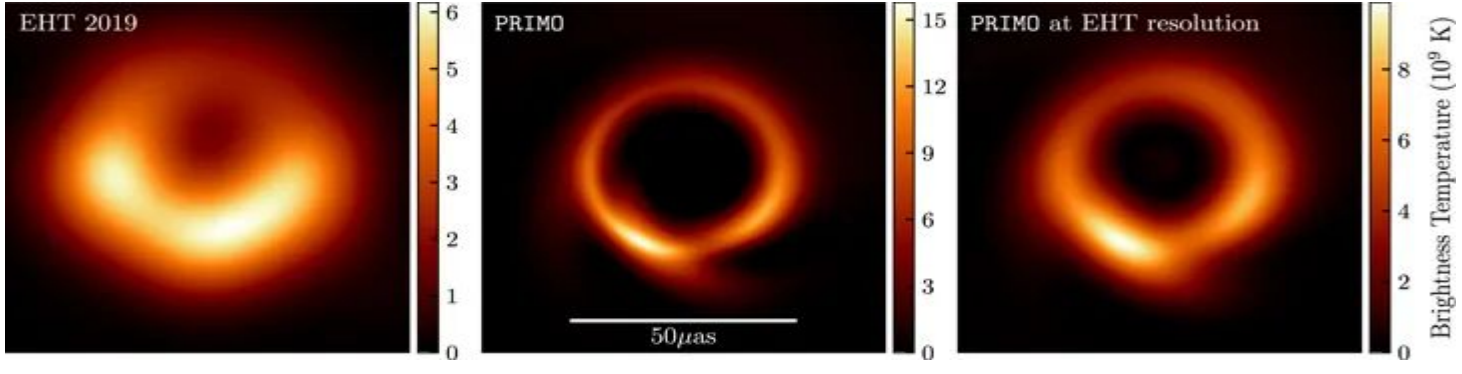
- We used AI to effectively analyze Cherenkov light, key for understanding neutrinos.

Enhanced Precision:

- By applying classification, image generation, and regression, we made data interpretation in neutrino detection more precise and versatile.

Broader Applications:

- These AI methods can also help tackle other astrophysical challenges, like analyzing black hole images from the EHT.



G

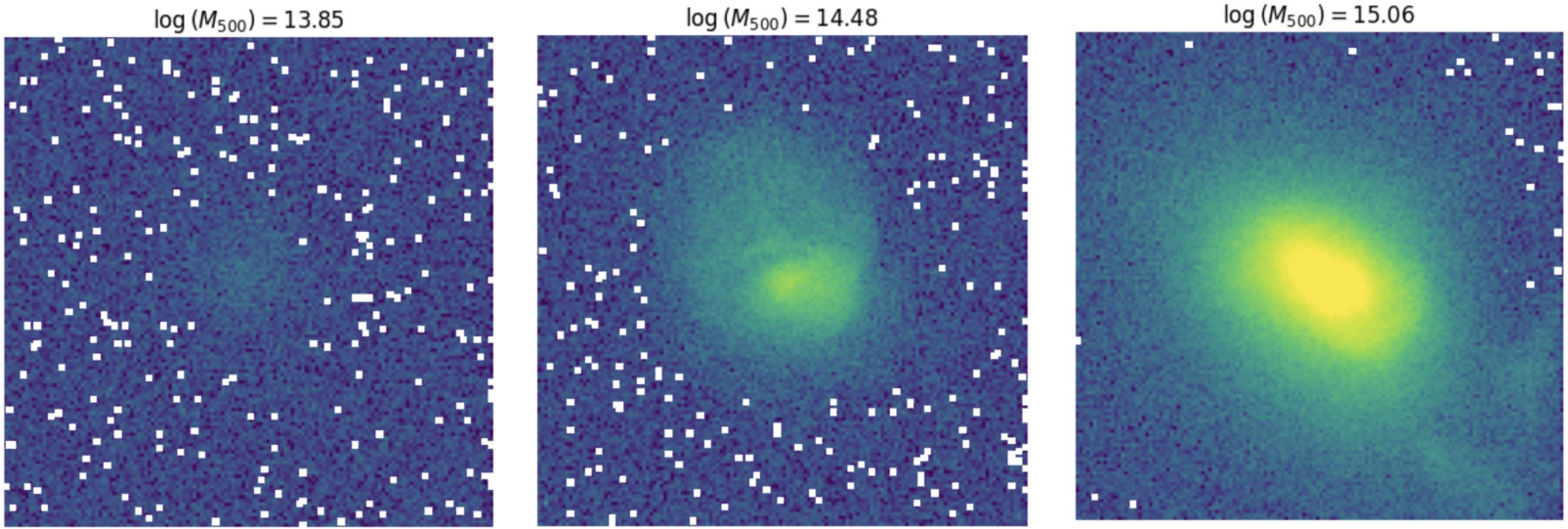
Project 2. Galaxy Cluster Masses with CNNs

Group G

Seyeon Hwang, Duho Kim, Xinru Li,Tomomi Sunayama,
Shota Takahashi, Jiashuo Zhang, Zhenghao Zhu

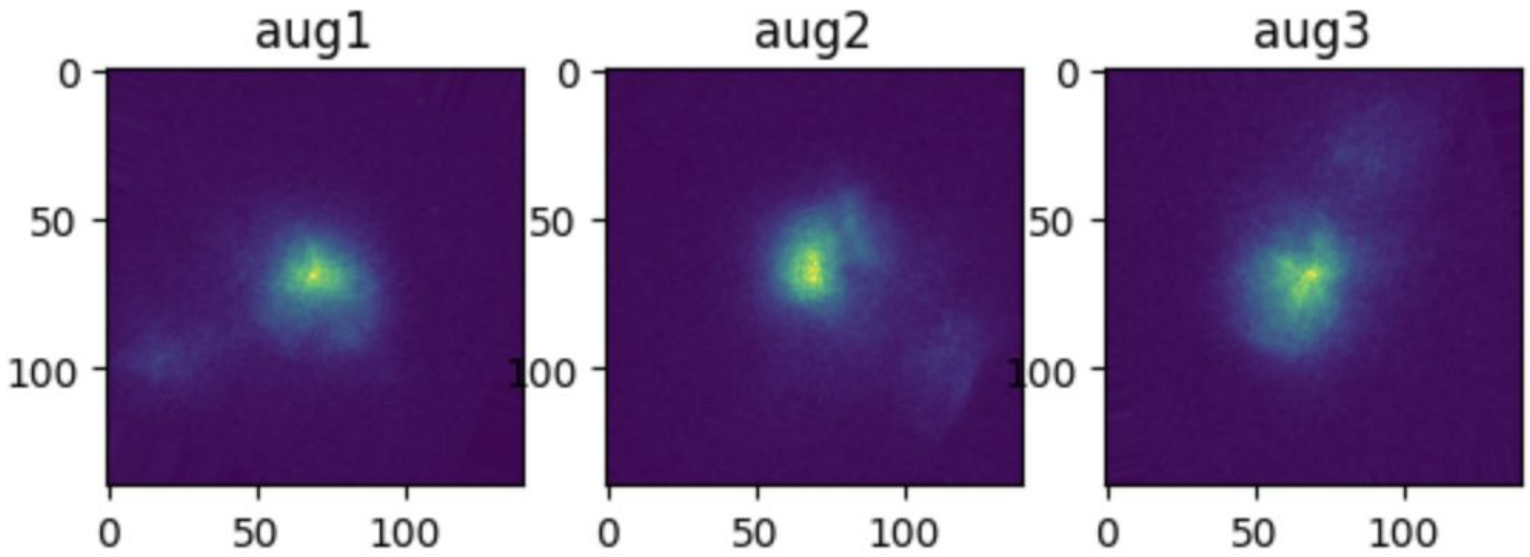
A3Net Summer School 2024

Images of galaxy clusters from TNG



low-mass → high-mass

Augmented Data (rotating 90 degree)



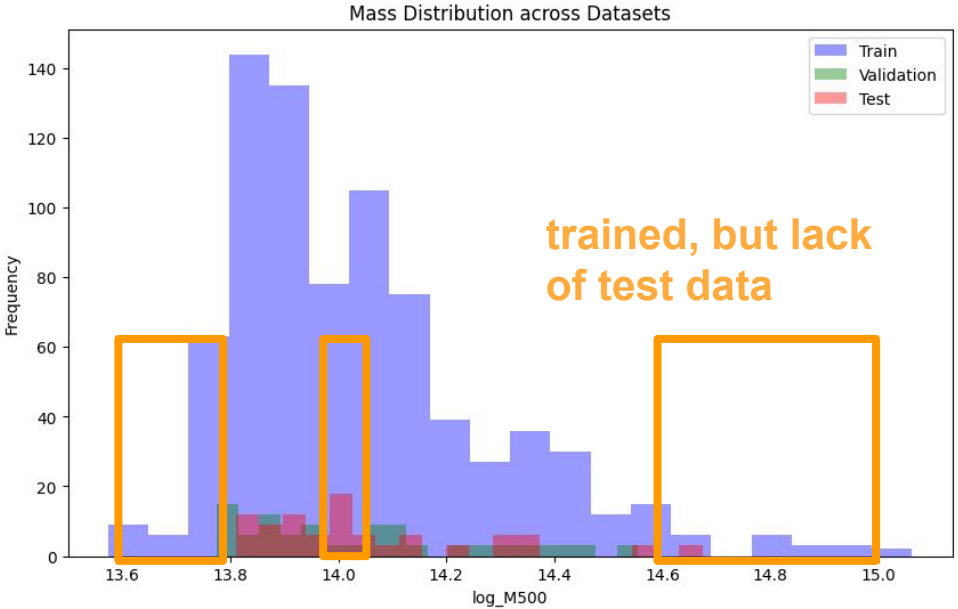
Datasets with Different Mass Distribution

Note: tuned model

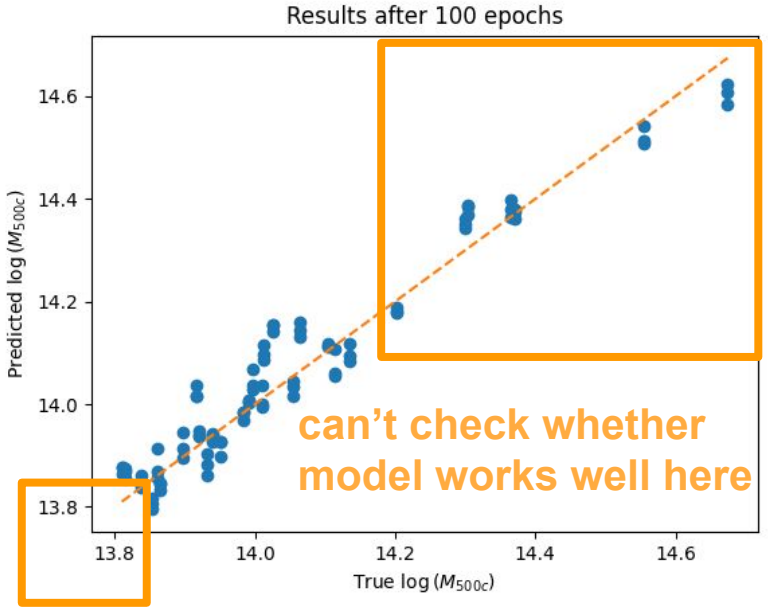
Dataset & Mass Distribution Bias?

Original Dataset: train : val : test = 794 : 99 : 93

Mass Distribution:



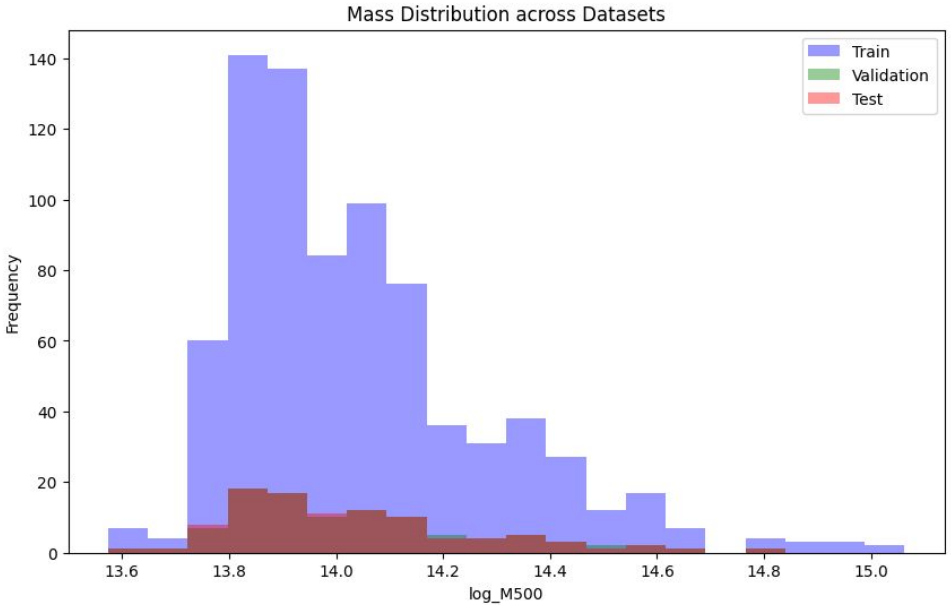
Result: Mean Squared Error = 0.00265



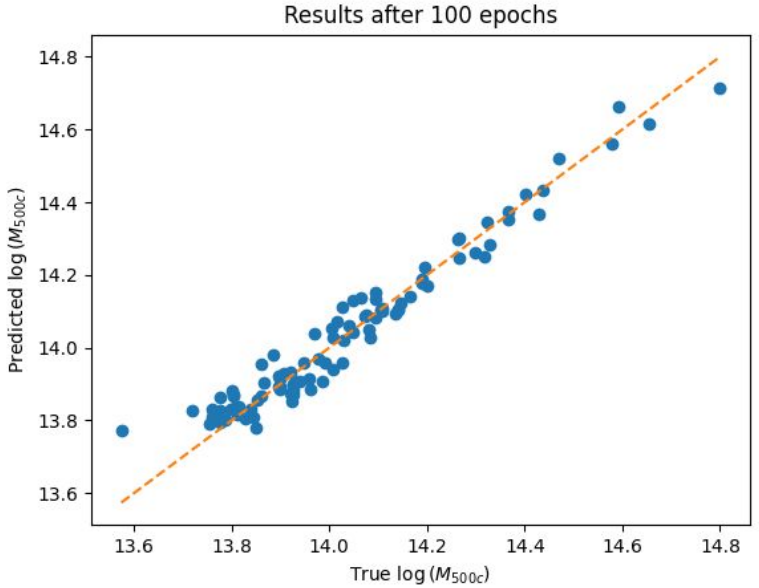
Shuffled Dataset: train : val : test = 788 : 99 : 99

use **Stratified Shuffle Split** to ensure train, val, test have similar mass distribution.

Mass Distribution:

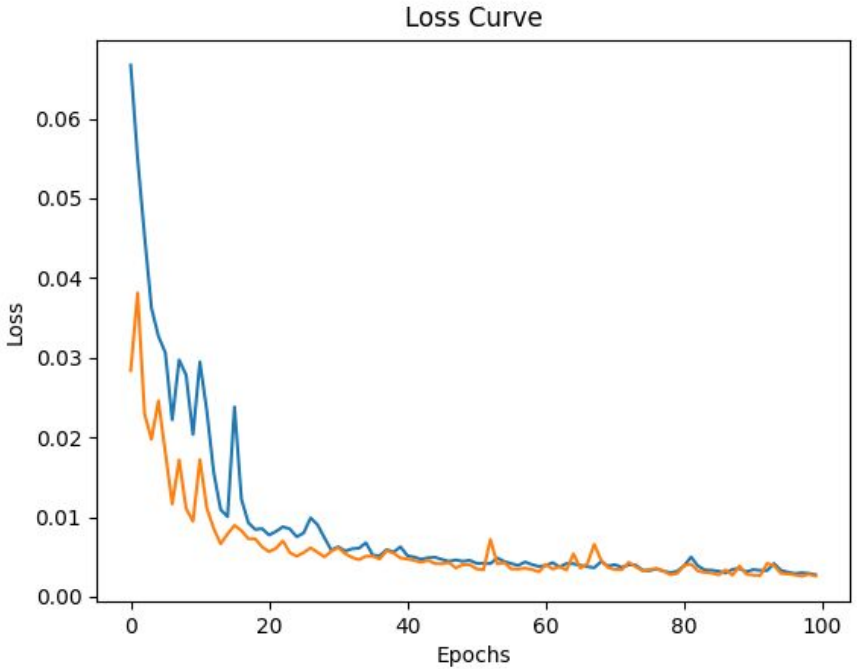


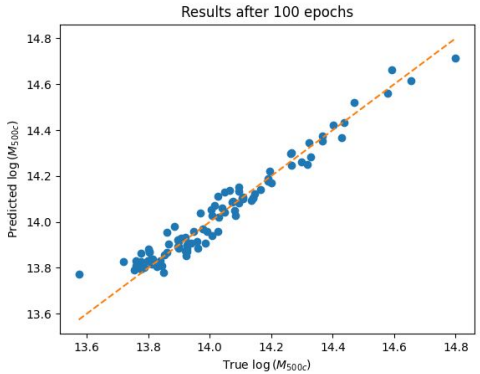
Result:



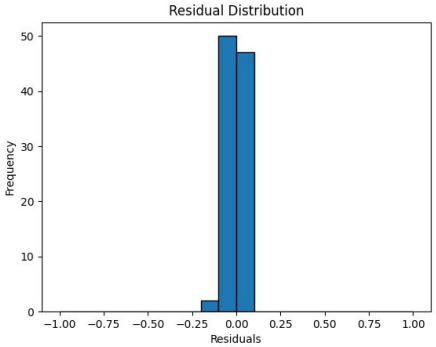
Shuffled Dataset: train : val : test = 788 : 99 : 99

Result:

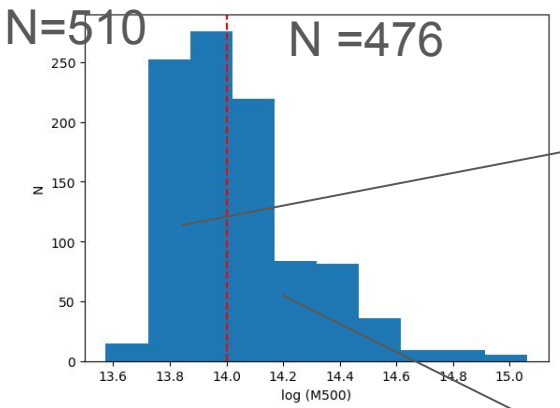


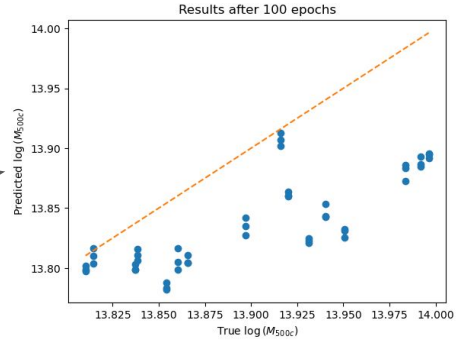


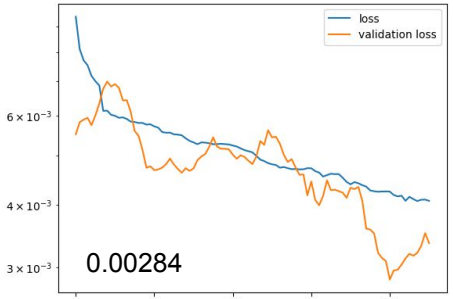
residuals = test Y - prediction

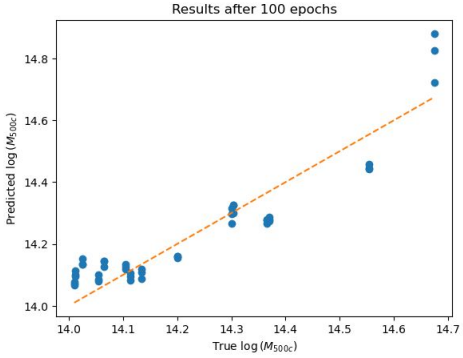


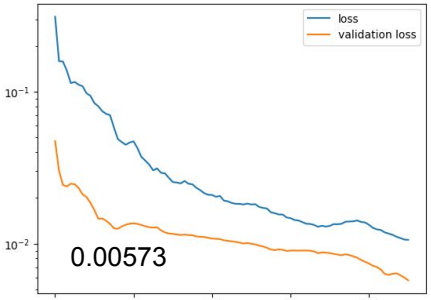
test loss: 0.0027
validation loss: 0.0026
Training completed in: 303.88 seconds
Mean Squared Error = 0.00242 < 0.00265 of original

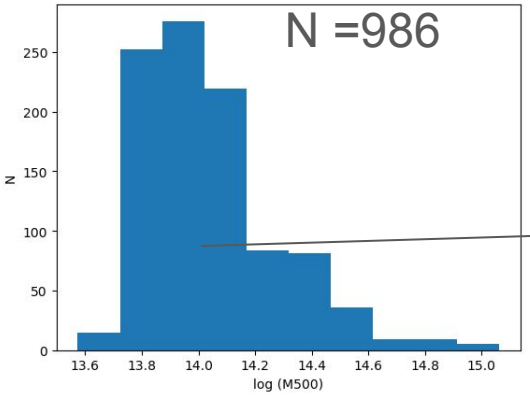

N=510
N =476
N
log (M500)

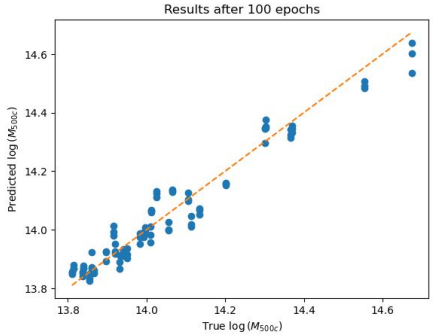

Results after 100 epochs
Predicted log (M_{500c})
True log (M_{500c})

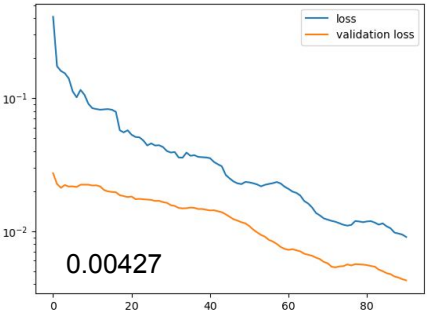

loss
validation loss
0.00284

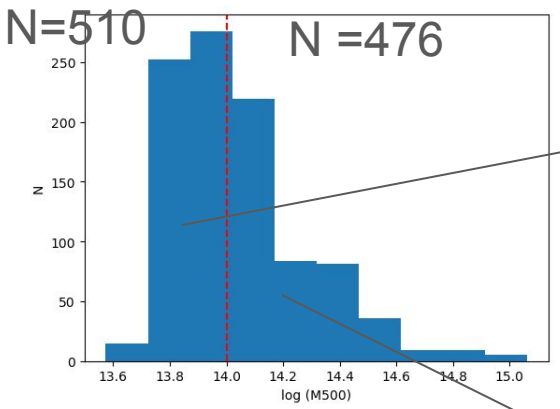

Results after 100 epochs
Predicted log (M_{500c})
True log (M_{500c})

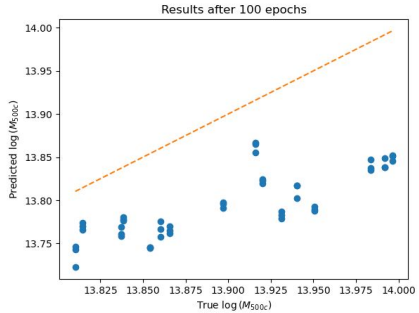

loss
validation loss
0.00573

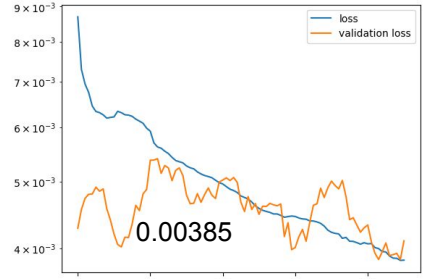

N =986
N
log (M500)


Results after 100 epochs
Predicted log (M_{500c})
True log (M_{500c})

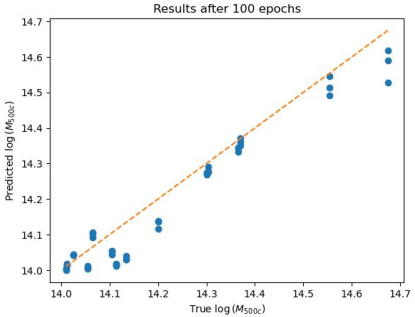

loss
validation loss
0.00427

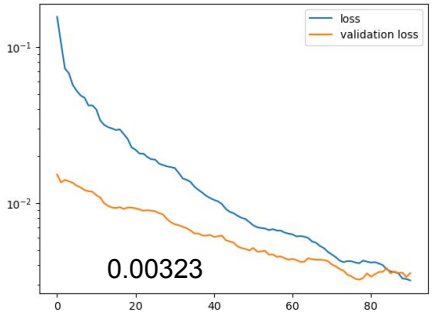


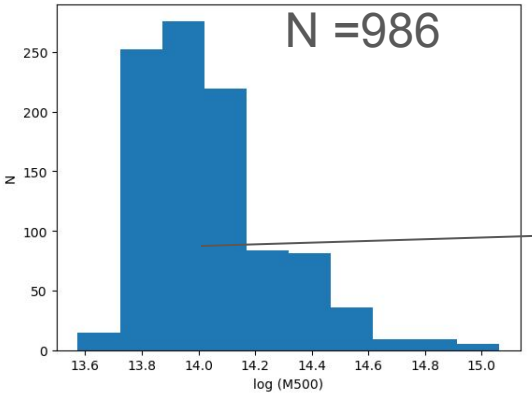


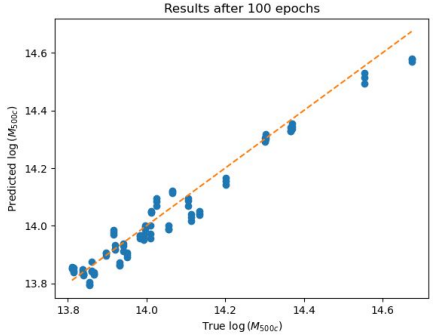


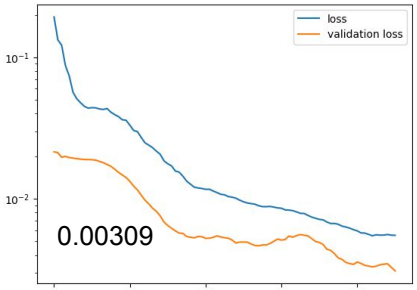
After augmenting training sample number to 1000 for all

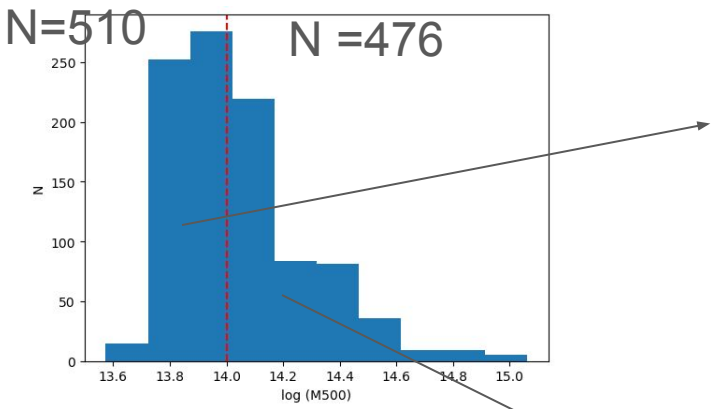


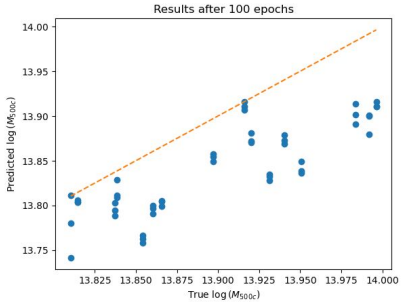


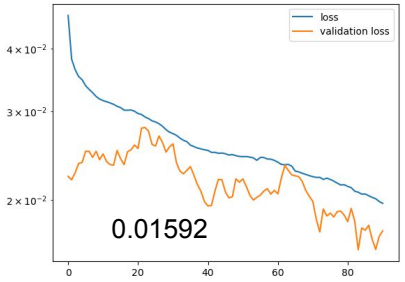




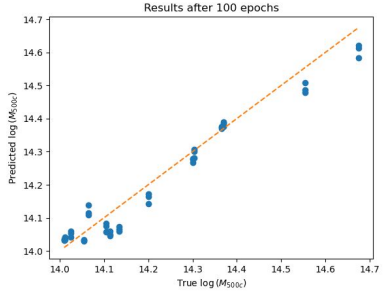


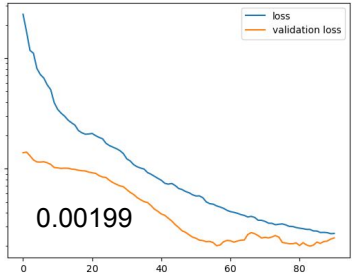


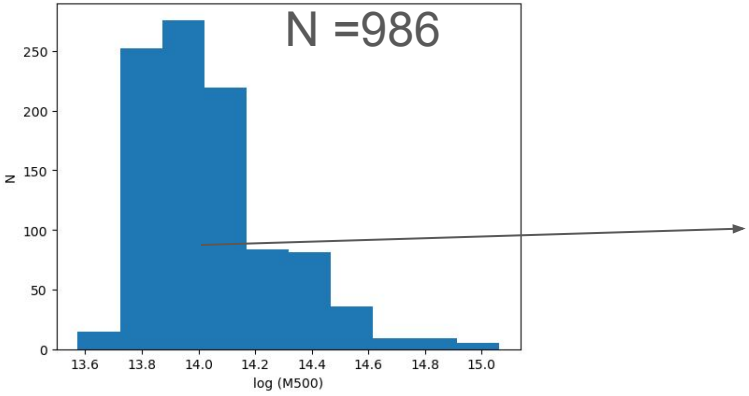


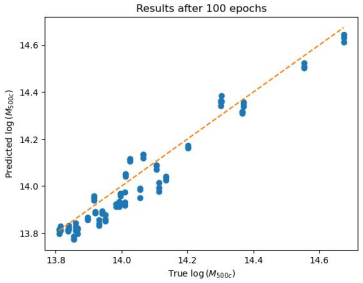


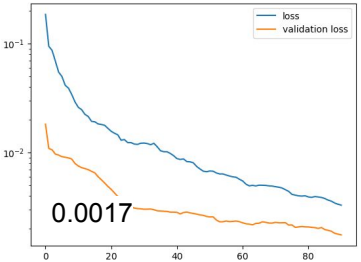
After normalizing the mass range to 0~1





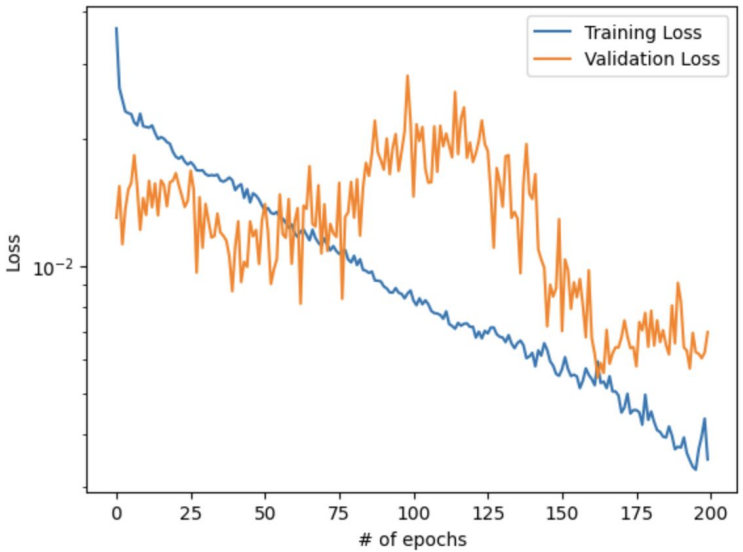


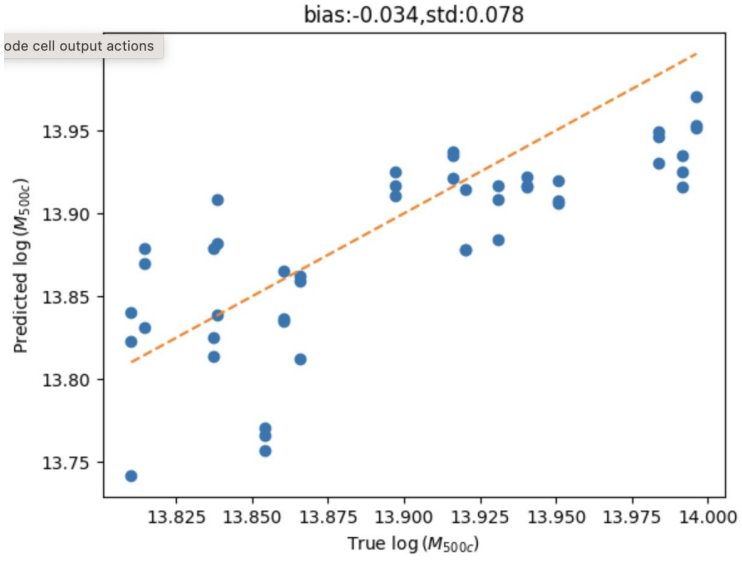




Tried with epoch=200 instead of epoch=100…

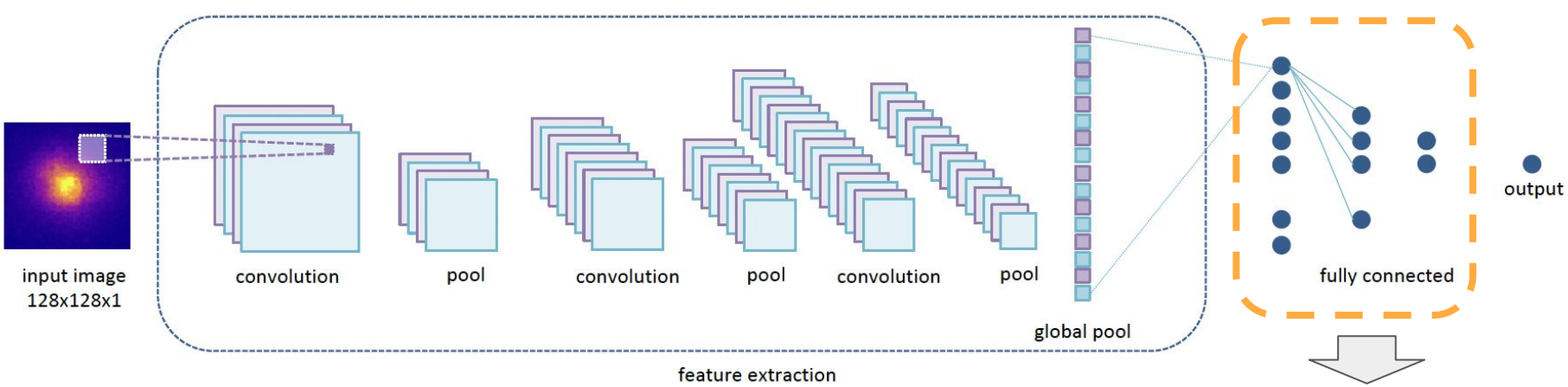
Tried with **all** the low-mass cluster sample: the subsample requires **more number of epochs** to converge and it gives a better result. However, the high-mass tail is still biased…





CNN Archtecture

Varying Hyperparameters



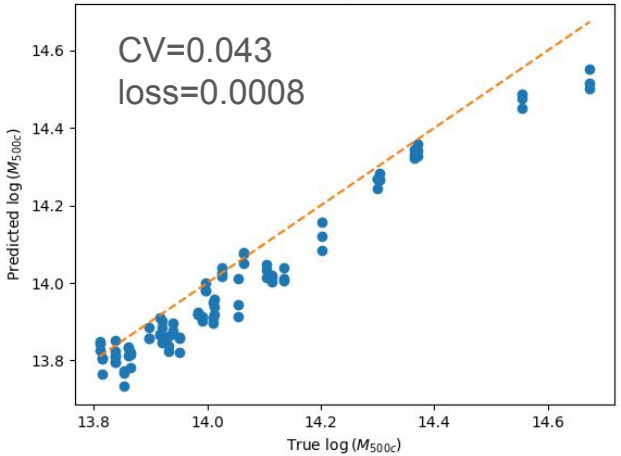
Double/half the number of fully-connected layers

Problem: Variations in the results are large enough to make conclusions because of small sample size (~1000) → use k-fold cross-validation

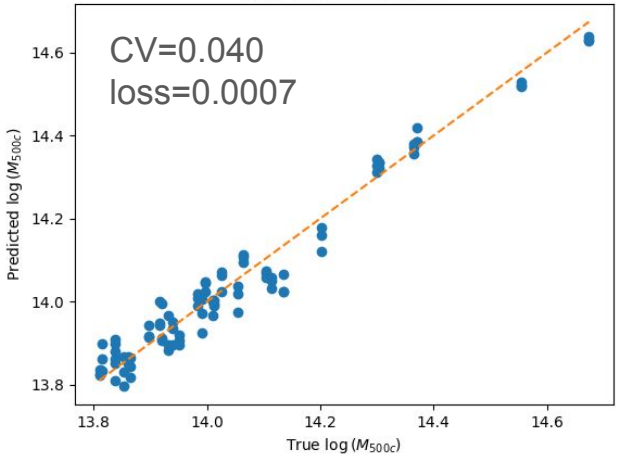
Varying Hyperparameters

By increasing the number of layers, the bias in the mass prediction becomes smaller (i.e., more accurate prediction).

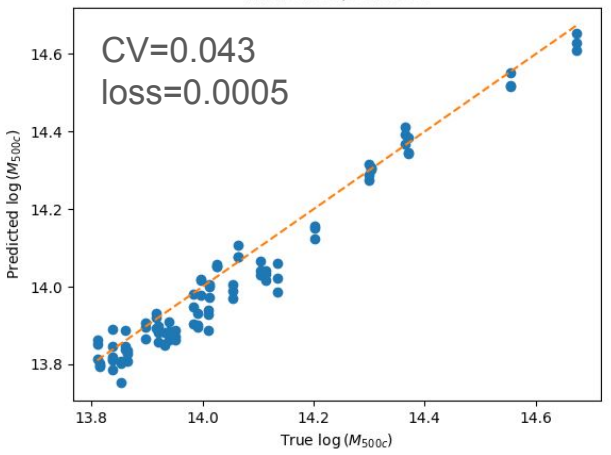
bias=-0.056



original
bias=-0.035



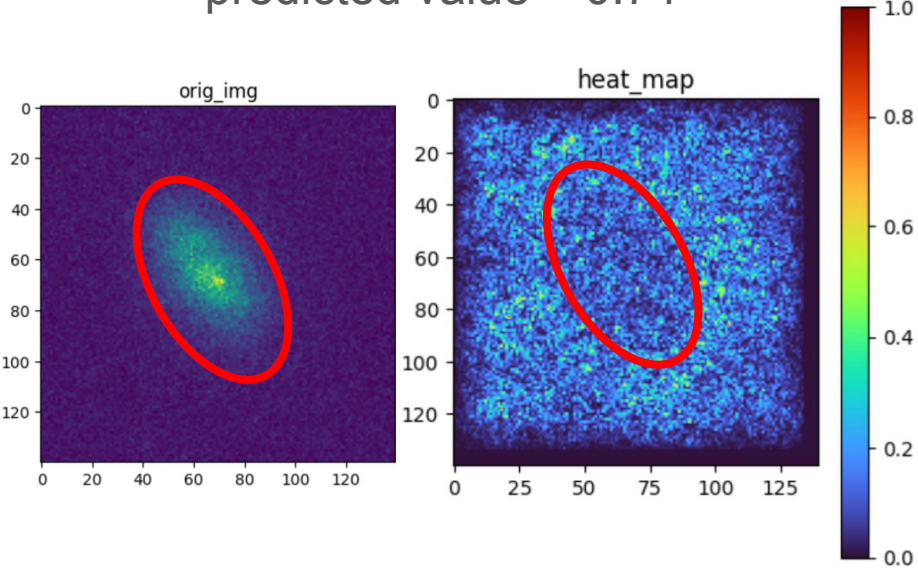
bias=-0.031



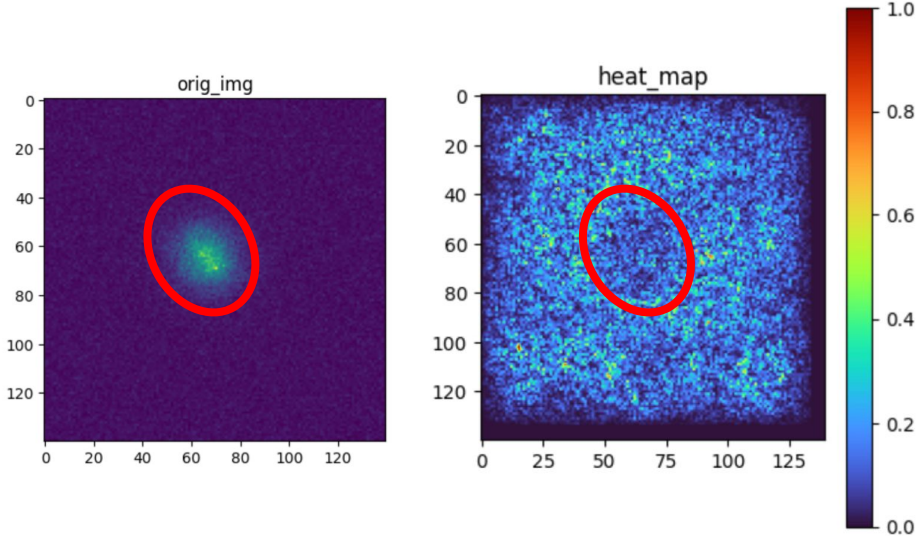
Small → Large

Saliency maps

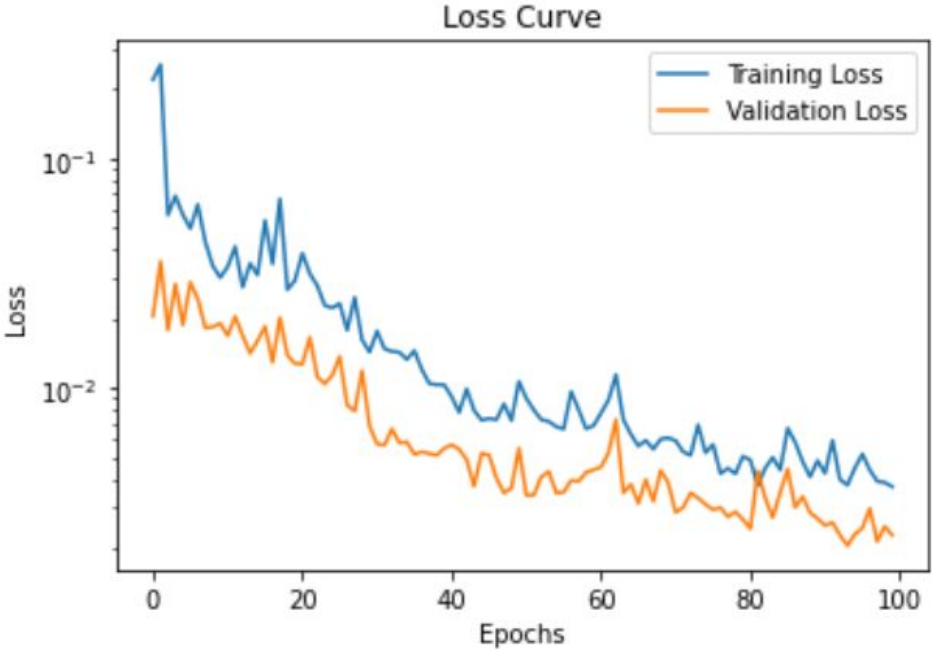
predicted value = 0.74

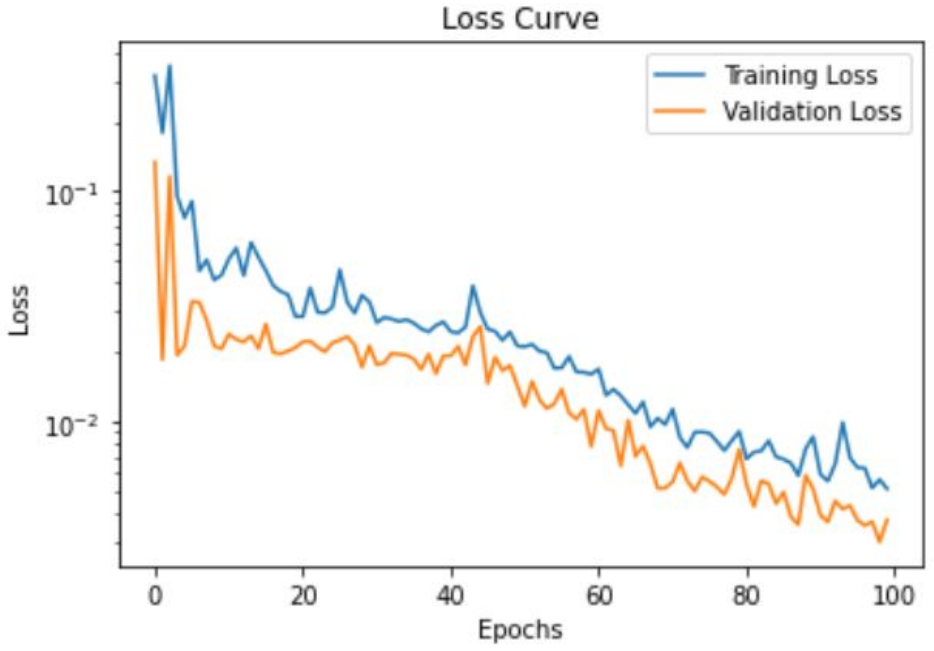


predicted value = 0.56



Rotational Invariance

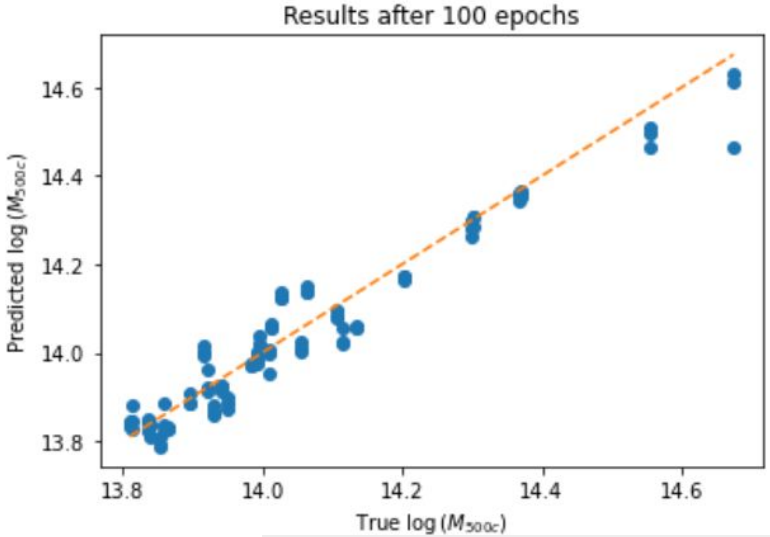


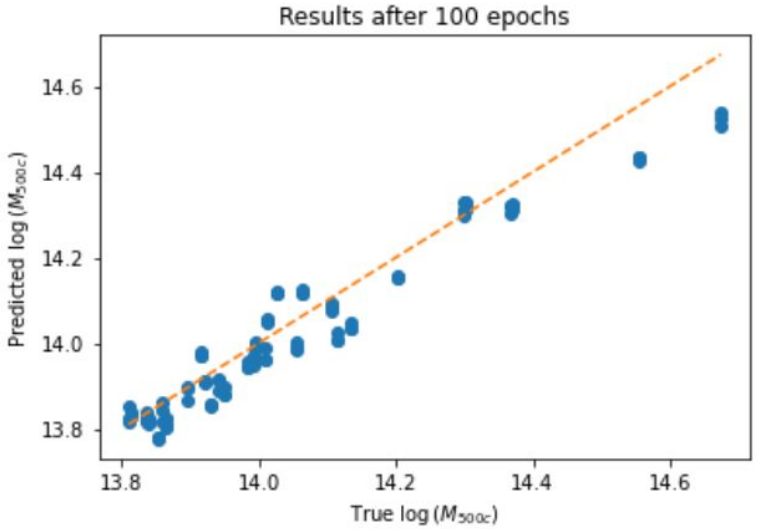


Left : Original model and original dataset

Right : apply random rotation to all training, validation, and test data set

BOTH with same hyperparameters 100 epoch





As long as we start from a ‘rotational invariant’ sample, the CNN trained is ‘rotational invariant’.

Left : Original model and original dataset

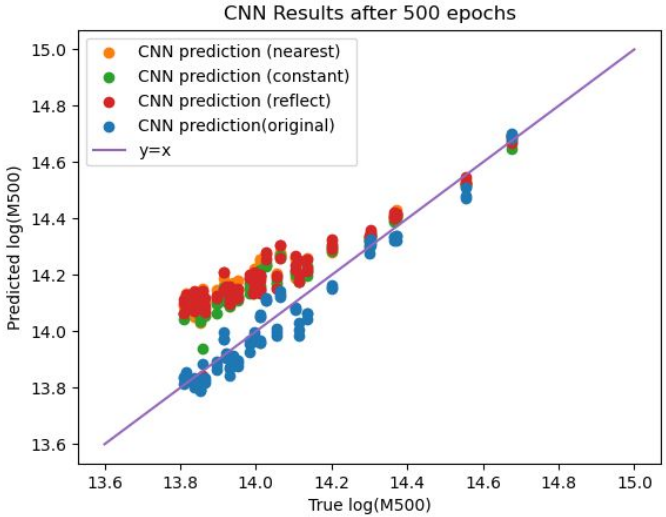
Right : apply random rotation to all training, validation, and test data set

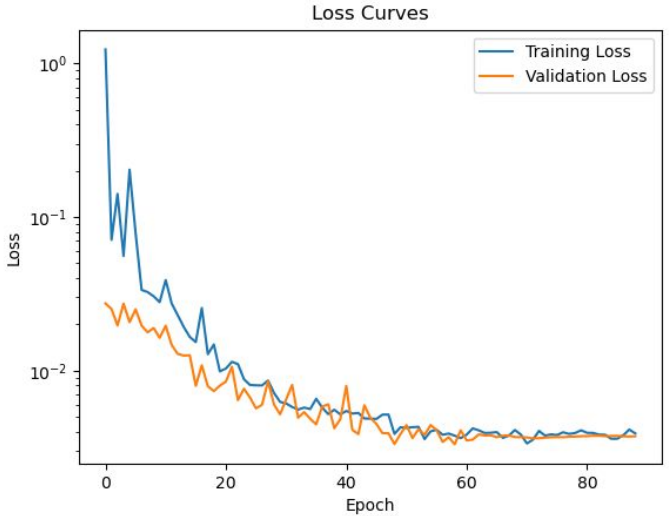
BOTH with same hyperparameters 100 epoch

Testing rotation performance on original CNN

loss on the original test data: ~ 0.002

loss on the **rotated test data**: 0.03-0.04





Rotation invariant network: gCNN model design

gCNN: Group Equivariant CNN (e2cnn package employed)

- 3 convolution layers followed by 3 pooling layers
- Group pooling layer
- mulit scale pooling
- fully conneted layer
- Dropout and Dense

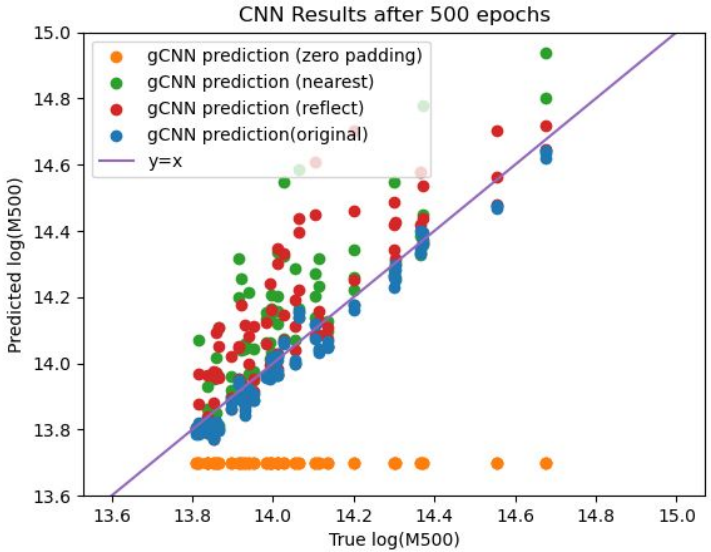
- Kernel designed on Rotation Group (Num_rotation=32)
- Multi scale feature extraction

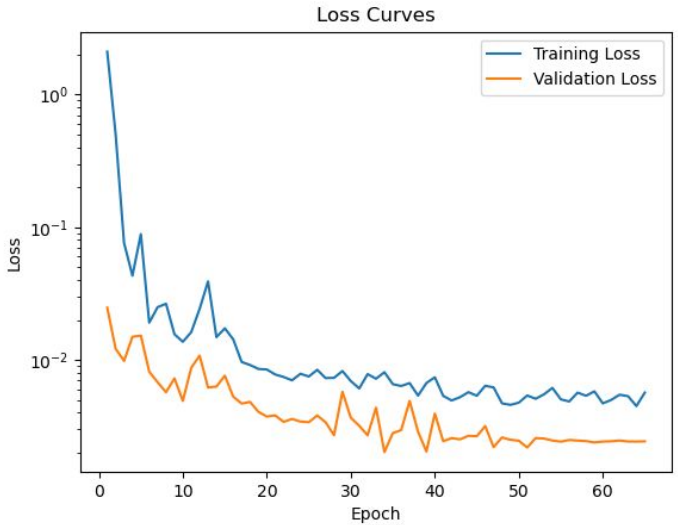
A very large model! (1.8G in size)

gCNN model result

loss on original test data (blue) ~0.002

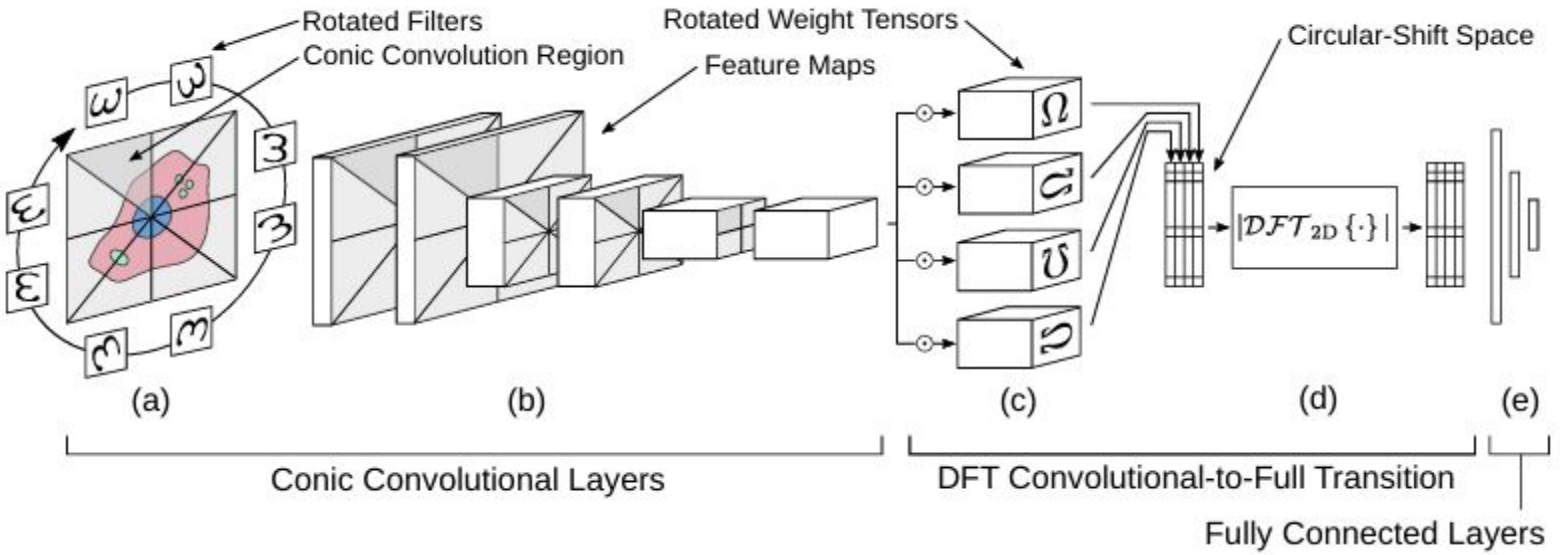
loss on **rotated test data**: 0.1(zero padding) 0.1(reflect) 0.06(nearest)





padding for the rotation? larger data set? better loss than MSE ?

RICNN


Rotated Filters
Conic Convolution Region
Rotated Weight Tensors
Feature Maps
Circular-Shift Space
$|\mathcal{DFT}_{2D}\{\cdot\}|$
(a)
(b)
(c)
(d)
(e)
Conic Convolutional Layers
DFT Convolutional-to-Full Transition
Fully Connected Layers

Summary

- Biased training data set could generate biased model (?), as seen in mass distribution and low-mass.

- The size of the dataset is crucial when exploring optimal architecture for CNN (i.e., data augmentation and k-fold cross-validation)

- ML considered the surrounding areas more important than the center of the cluster.

- To achieve rotational invariance in CNN, rotation needs to be applied somewhere. Either on original data-set or on the convolution kernels.

APPENDIX

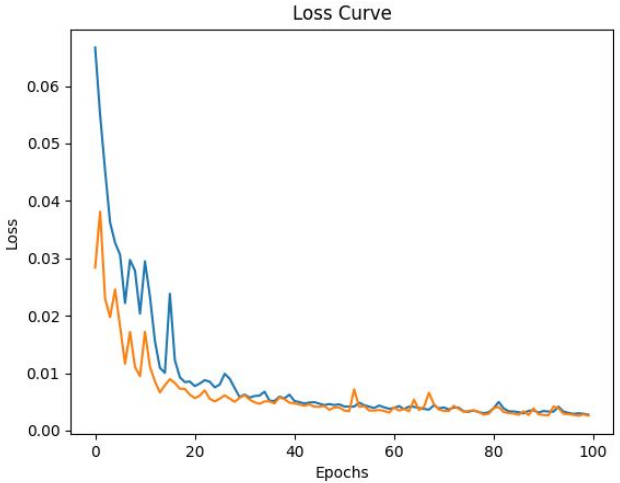
Model: "sequential_4", CNNs

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D(32, (3,3), relu))	(None, 138, 138, 32)	320
max_pooling2d_12 (MaxPooling2D(2,2))	(None, 69, 69, 32)	0
conv2d_13 (Conv2D(64, (3,3), relu))	(None, 67, 67, 64)	18496
max_pooling2d_13 (MaxPooling2D(2,2))	(None, 33, 33, 64)	0
conv2d_14 (Conv2D(64, (3,3), relu))	(None, 31, 31, 64)	36928
max_pooling2d_14 (MaxPooling2D(2,2))	(None, 15, 15, 64)	0
global_average_pooling2d_4 (GlobalAveragePooling2D)	(None, 64)	0
dropout_8 (Dropout(0.1))	(None, 64)	0
dense_16 (Dense(200, relu))	(None, 200)	13000
dropout_9 (Dropout(0.1))	(None, 200)	0
dense_17 (Dense(100, relu))	(None, 100)	20100
dense_18 (Dense(20, relu))	(None, 20)	2020
dense_19 (Dense(1,softpuls))	(None, 1)	21

Total params: 90,885

Trainable params: 90,885

Non-trainable params: 0



H

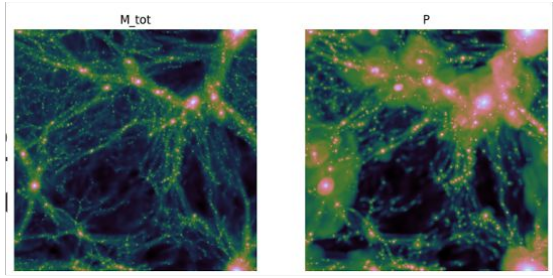
Group H

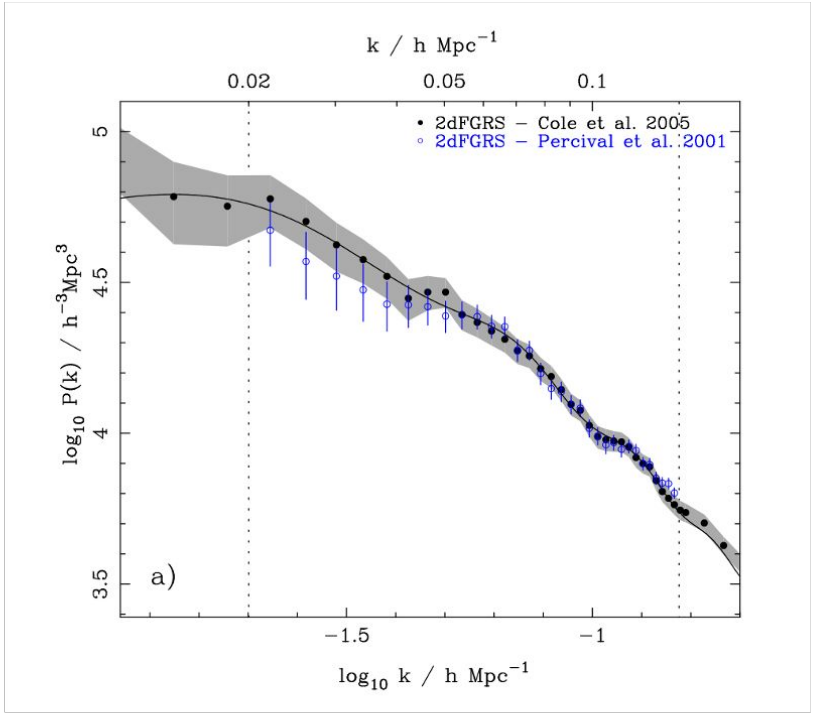
(**8**) Using **CNNs** to infer the fundamental **parameters** of the Universe

Chenze Dong / Jiacheng Ding / Mingyeong Yang / Shinsei Eyama / Xinyue Chen / YoungPyo Hong / Yiming Dong

Motivation

- How to determine Ω_m and σ_8 from large scale survey?
  - The main point is to compare observations with theoretical calculations or simulations
  - But, we have no idea how to compare 2 dimensional figure directly
  - A possible method or compromise is data dimension reduction, such as with power spectrum, which may cause information loss

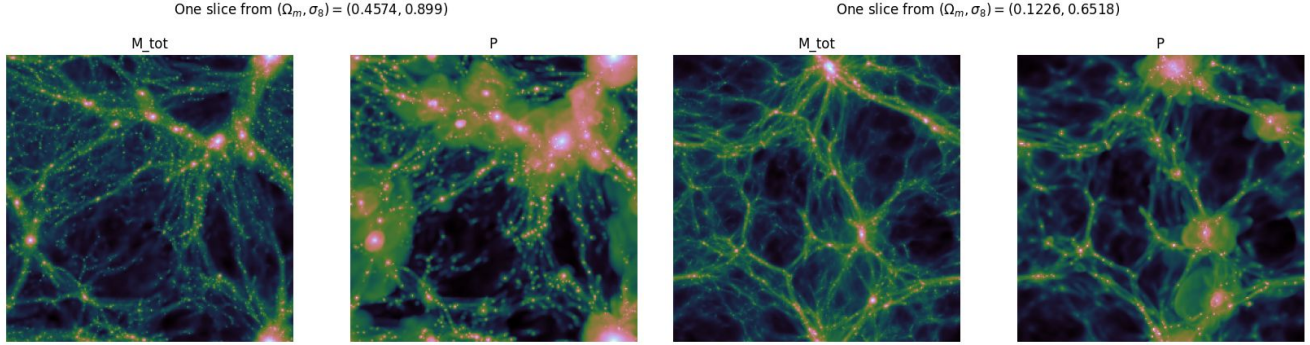


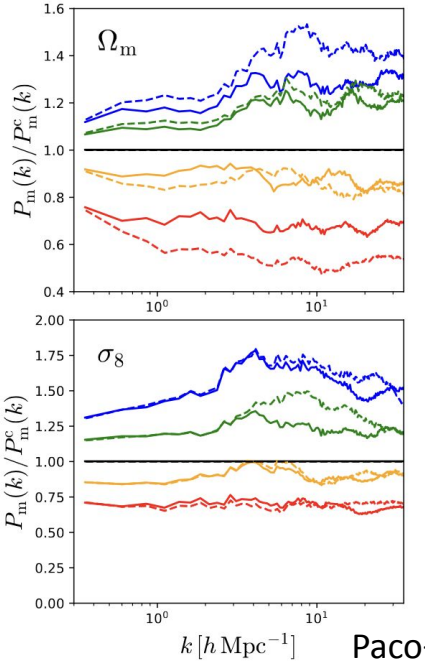


2dFGRS Collaboration, 2005

Our “raw” dataset (from **TNG**)

- **2D image** of **M_tot** (total mass) & **P** (electron pressure)
- **1000 sets** of cosmological / astrophysical **parameters** from LH sampling
  - Cosmological param. (**Ω_m, σ_8**)
- 15 slices x 1000 settings = **15000 images**
  - with size (2,256,256)

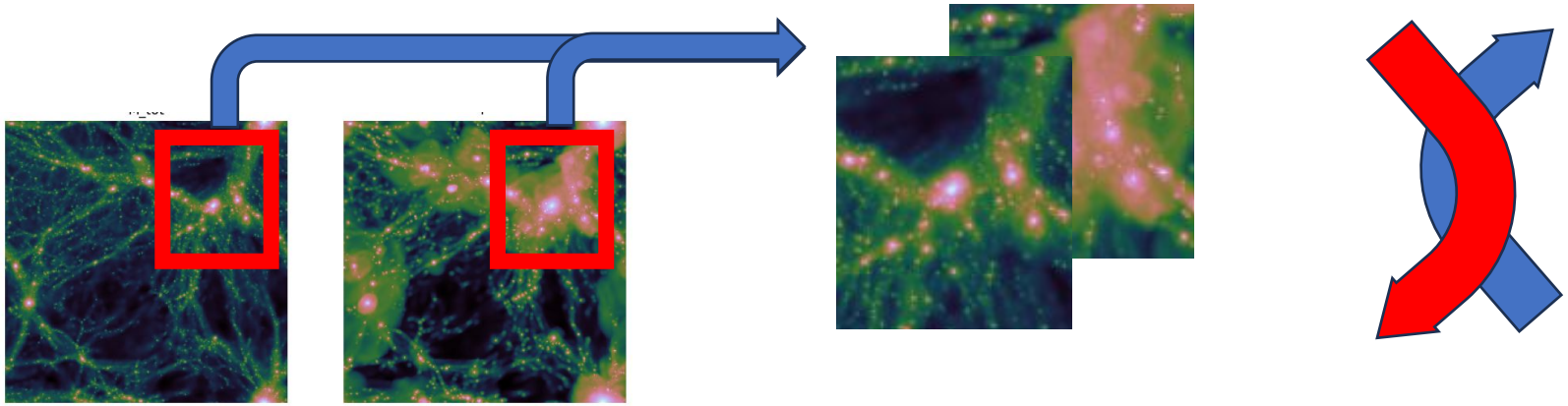




Paco+20

Preprocessing

- Strategies:
  - **Combine** the two maps together as two channels of one image
  - Work with **log**-image & **normalization** to [0, 1]
  - Crop the image to a **smaller** size
  - **Shuffle** images so different cosmologies can enter the training set



Simple model

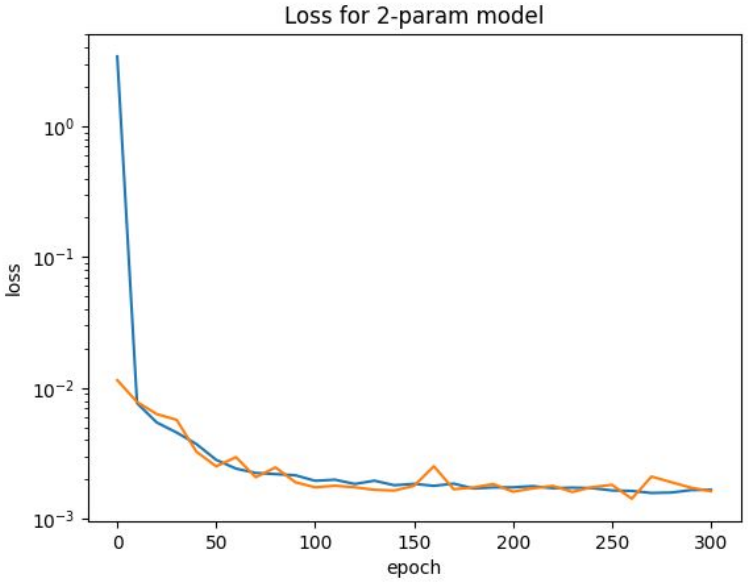
- ***Simple* CNN model**
  - Taking input of (*,2,64,64) image
  - 3 layers of Convolution + MaxPooling
  - Layer normalization for each layer
  - 2 fully connected layers
  - Giving output two cosmo. param. (*,2)
  - Training for 300 epochs

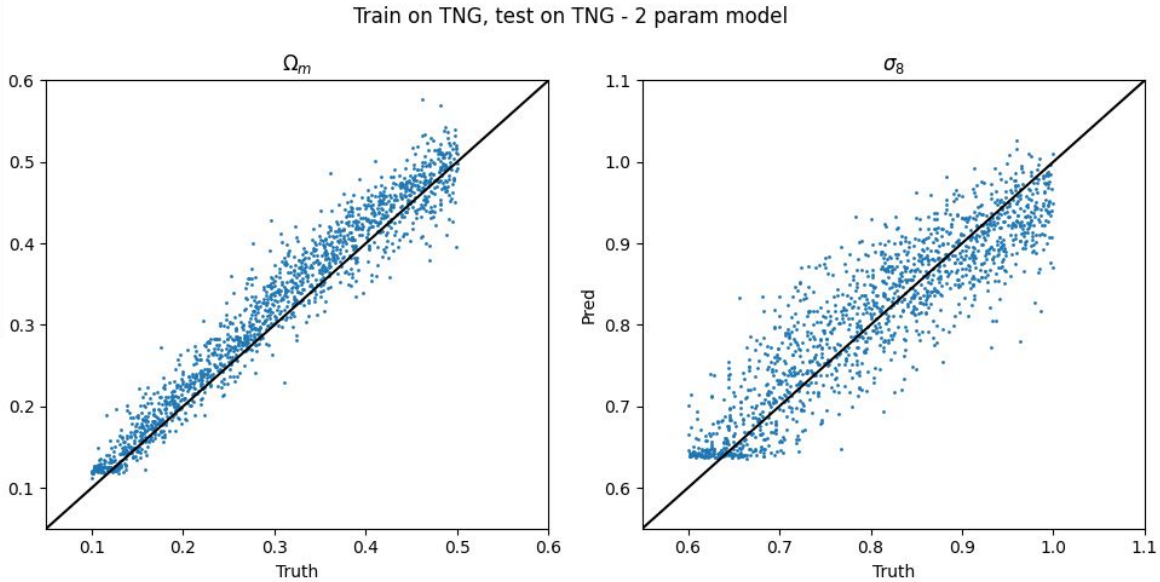
Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 64, 64]	304
ReLU-2	[-1, 16, 64, 64]	0
LayerNorm-3	[-1, 16, 64, 64]	131,072
MaxPool2d-4	[-1, 16, 32, 32]	0
Conv2d-5	[-1, 32, 32, 32]	4,640
ReLU-6	[-1, 32, 32, 32]	0
LayerNorm-7	[-1, 32, 32, 32]	65,536
MaxPool2d-8	[-1, 32, 16, 16]	0
Conv2d-9	[-1, 64, 16, 16]	18,496
ReLU-10	[-1, 64, 16, 16]	0
LayerNorm-11	[-1, 64, 16, 16]	32,768
MaxPool2d-12	[-1, 64, 8, 8]	0
Linear-13	[-1, 1024]	4,195,328
ReLU-14	[-1, 1024]	0
Linear-15	[-1, 2]	2,050

Total params: 4,450,194
Trainable params: 4,450,194
Non-trainable params: 0

Input size (MB): 0.03
Forward/backward pass size (MB): 2.86
Params size (MB): 16.98
Estimated Total Size (MB): 19.87

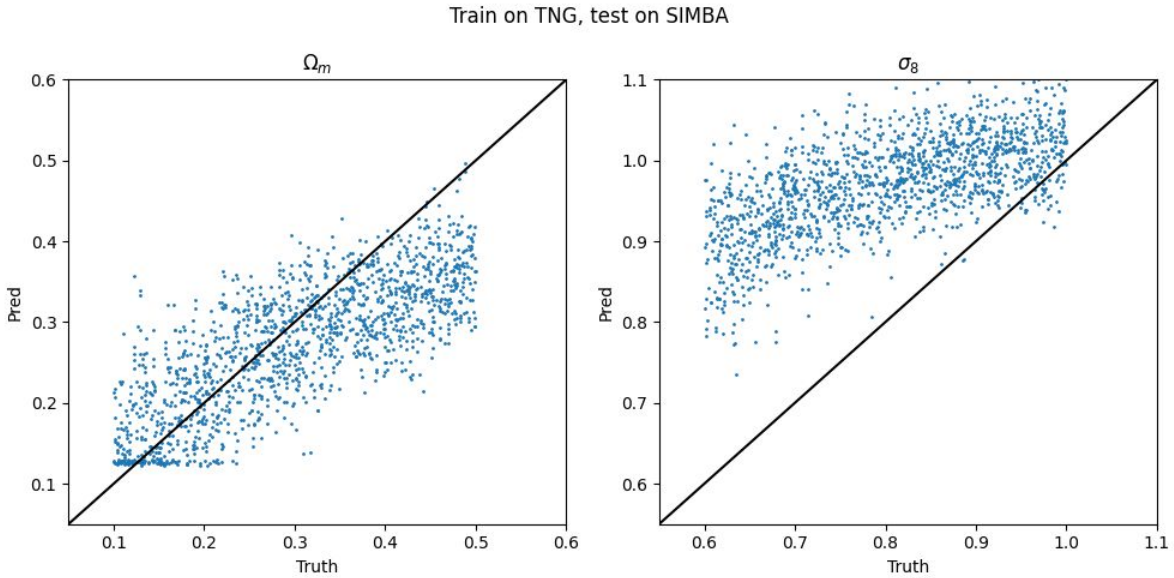
Simple model, good performance

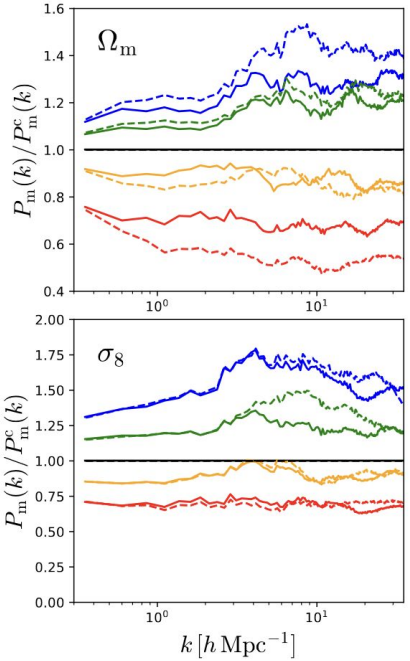




Does the model generalize?

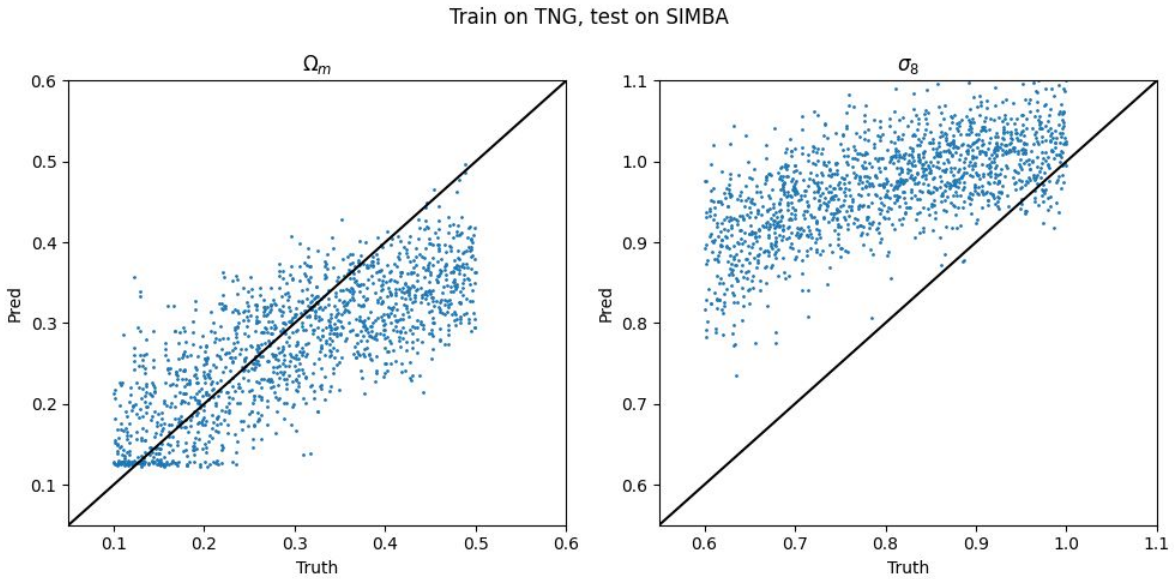
- Not that bad on **Ω_m**, overestimated for **σ_8**

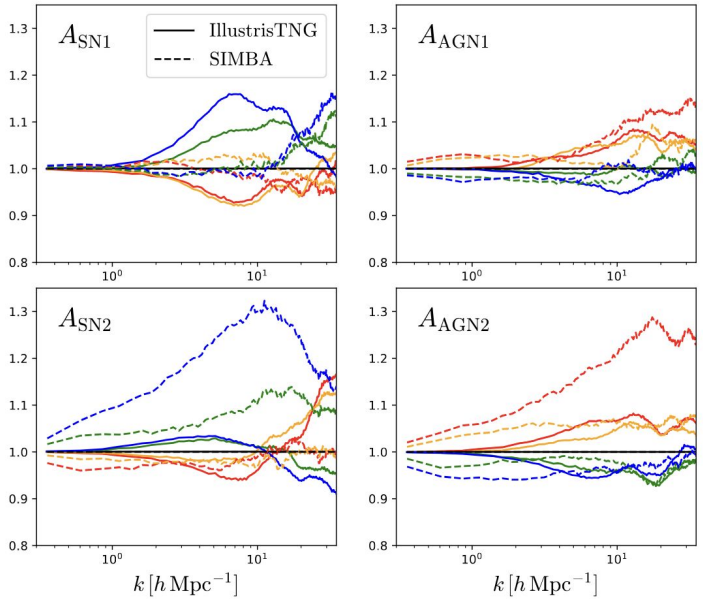




Does the model generalize?

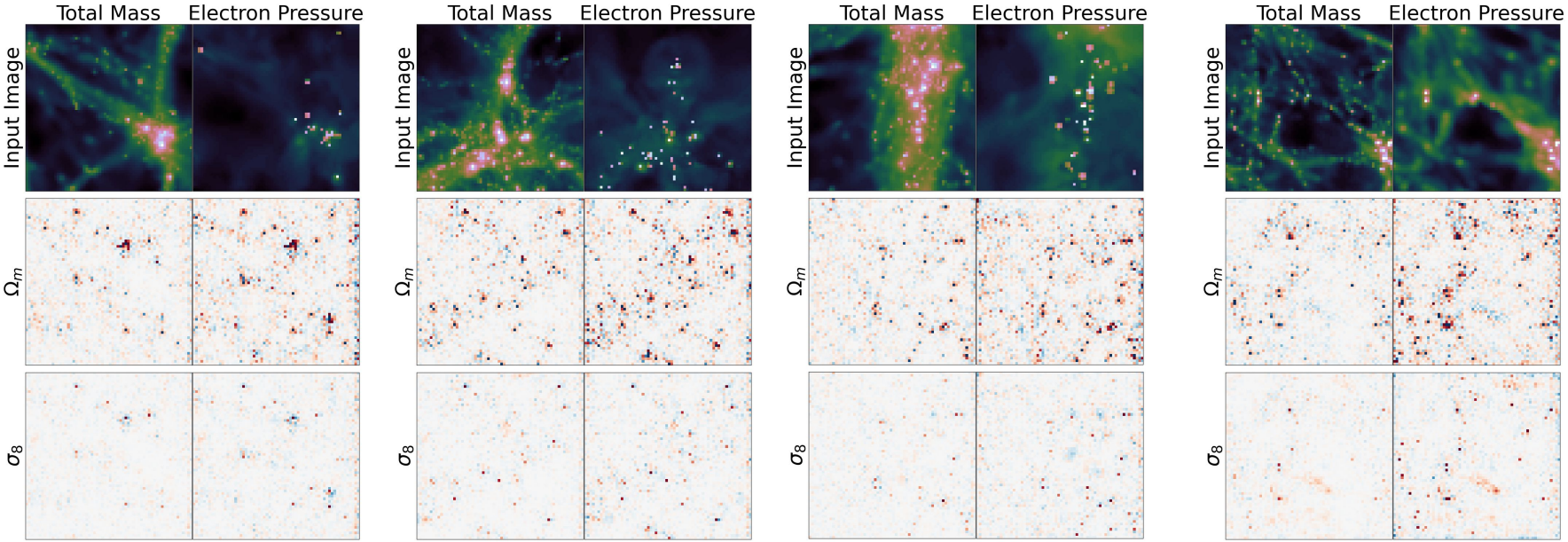
- Not that bad on **Ω_m**, overestimated for **σ_8**





Interpret this CNN

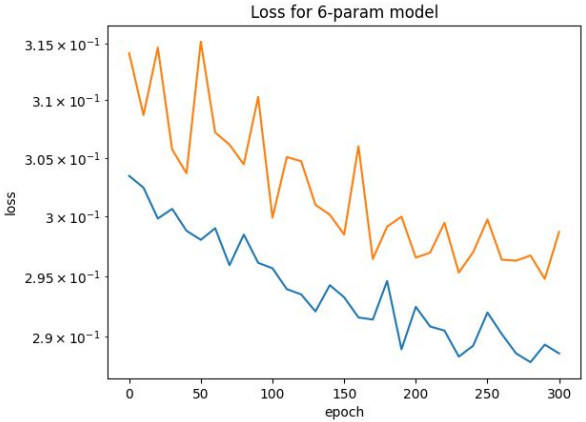
- The network is **NOT** telling cosmology from **massive halos**
- It seemingly uses the information from value peaks in **filaments**

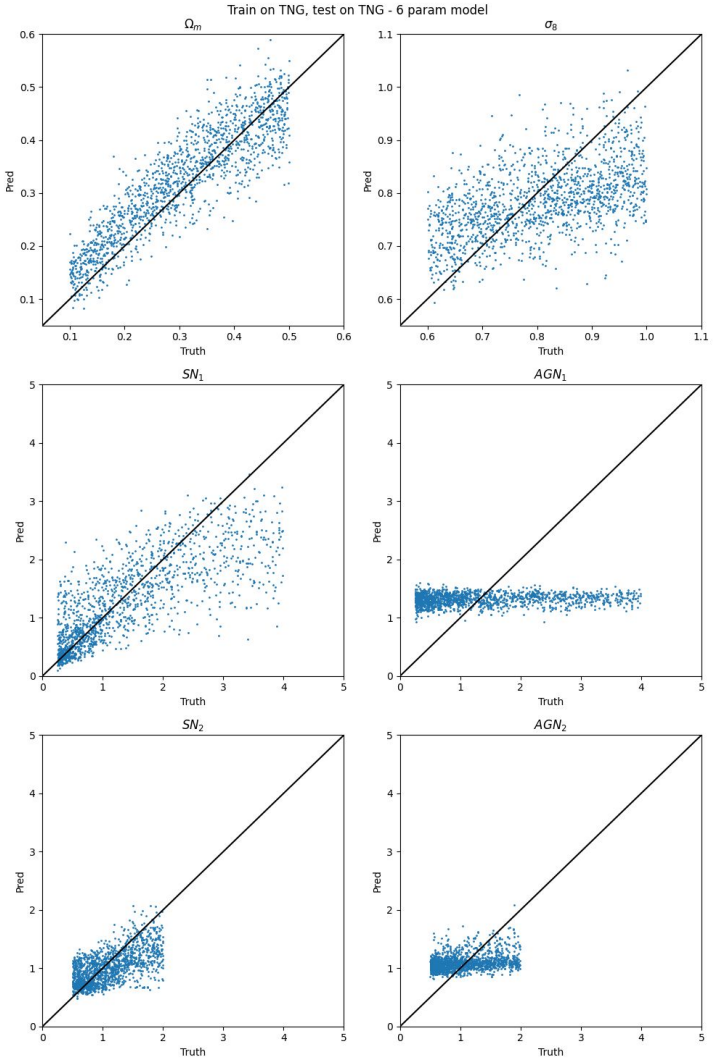


Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 64, 64]	304
ReLU-2	[-1, 16, 64, 64]	0
LayerNorm-3	[-1, 16, 64, 64]	131,072
MaxPool2d-4	[-1, 16, 32, 32]	0
Conv2d-5	[-1, 32, 32, 32]	4,640
ReLU-6	[-1, 32, 32, 32]	0
LayerNorm-7	[-1, 32, 32, 32]	65,536
MaxPool2d-8	[-1, 32, 16, 16]	0
Conv2d-9	[-1, 64, 16, 16]	18,496
ReLU-10	[-1, 64, 16, 16]	0
LayerNorm-11	[-1, 64, 16, 16]	32,768
MaxPool2d-12	[-1, 64, 8, 8]	0
Linear-13	[-1, 4096]	16,781,312
ReLU-14	[-1, 4096]	0
Linear-15	[-1, 6]	24,582

Total params: 17,058,710
Trainable params: 17,058,710
Non-trainable params: 0

Input size (MB): 0.03
Forward/backward pass size (MB): 2.91
Params size (MB): 65.07
Estimated Total Size (MB): 68.01





Can you infer 6 parameters? No:(

Summary

- We constructed a simple, **3-layer CNN** to infer (**Ω_m, σ_8**) from (M_tot, P) slices of **TNG** suites in **CAMELS** simulation.

- The **Result** is actually **good** given the simple architecture; when testing with **SIMBA** data, the model more or less **generalizes** for **Ω_m,** but usually **overestimate σ_8**.
- By inspecting the saliency map, we find the **model** is likely extracting **information** from **halos** in filaments rather than **cosmic web nodes**.
- We also attempt to train a **similar model** to predict **all** the **6** simulation **parameters**. After 300 epoches' training, the model learns almost **nothing** about **AGN feedback**, suggesting a **minor impact** of AGN feedback on the **two input** quantities.