Estimating Photometric Galaxy Redshifts from Machine Learning

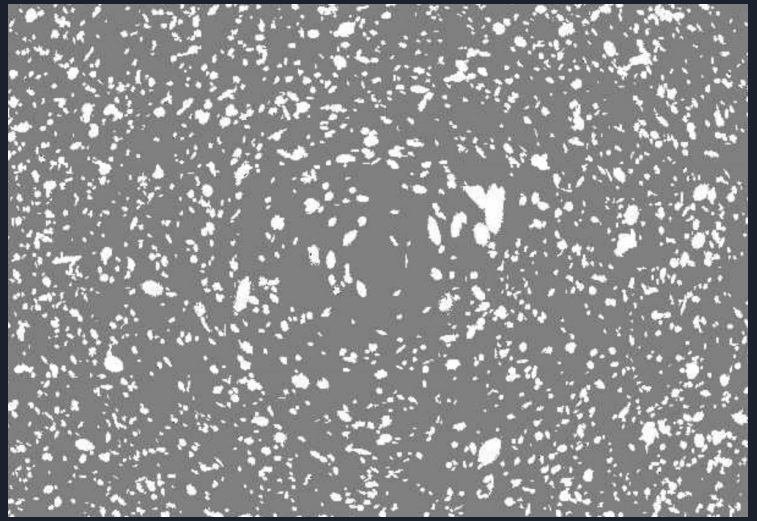
F. Deng, J. Ducharme, Z. Escalante, S. Helhoski, S. Yan

Introduction

The Astrolensing Group at Brown currently focuses heavily on using gravitational lensing to make predictions about cosmological parameters.

Gravitational lensing occurs when the path of light from a background object is distorted by a massive foreground object, causing irregularities in the magnification and shape of the background objects.

We can reverse engineer the mass in the foreground if we know the angular locations, shapes, and redshifts of the background objects.



Weak Lensing
(background objects)

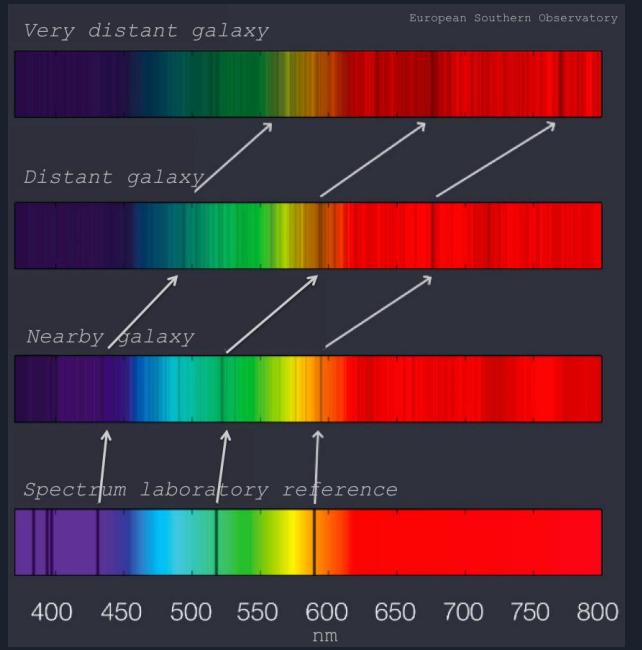
Introduction

Redshift is a measure of the distance to a cosmological object, denoted by z

Since the speed of light is finite, light from galaxies takes time to reach us. During that time, the universe expands, causing wavelengths to stretch.

Spectroscopic redshift (spec-z): The most accurate means of measuring z, found by observing light from a galaxy, splitting light through a prism, identifying component wavelengths, and comparing with rest wavelengths.

$$z = \frac{\Delta\lambda}{\lambda_{rest}} = \frac{\lambda_{obs} - \lambda_{rest}}{\lambda_{rest}} = \frac{v}{c}$$

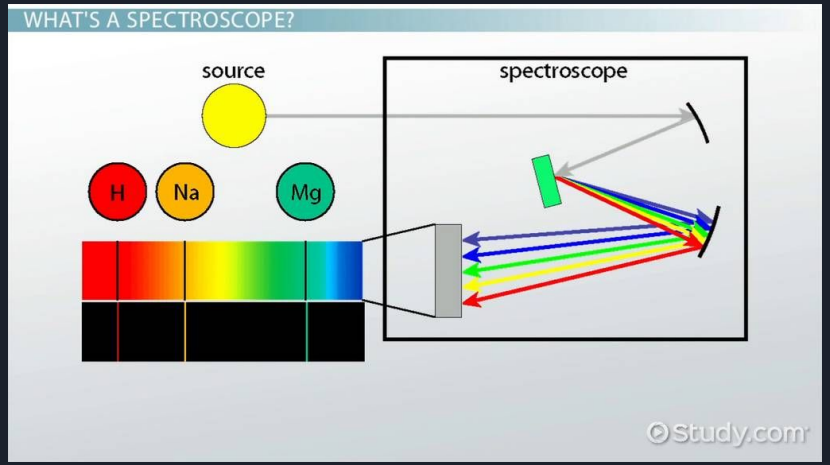


https://itu.physics.uiowa.edu/labs/advanced/astronomical-redshift

Introduction

Spectroscopy is considered a very reliable source of redshift information. Redshift values predicted in this way are called spectroscopic redshifts.

We use the celestial coordinates of each galaxy in our data to look up known values of spectroscopic redshift from the NASA/IPAC Extragalactic Database



https://study.com/learn/lesson/spectroscope-diagram-parts-function.html

Why Machine Learning

How can we predict the redshift of a galaxy that is not in the NED database?

Why Machine Learning

How can we predict the redshift of a galaxy that is not in the NED database?

Use the flux measurements of a galaxy in different bands of light.
(Known as Photometric Redshift)

Essentially we use a discrete spectrum of light that is captured by taking optical images through multiple filters.

In our case we will let machine learning take care of constructing the model, rather than constructing it manually ourselves.

We can train on galaxies that also have spectroscopic redshift information.

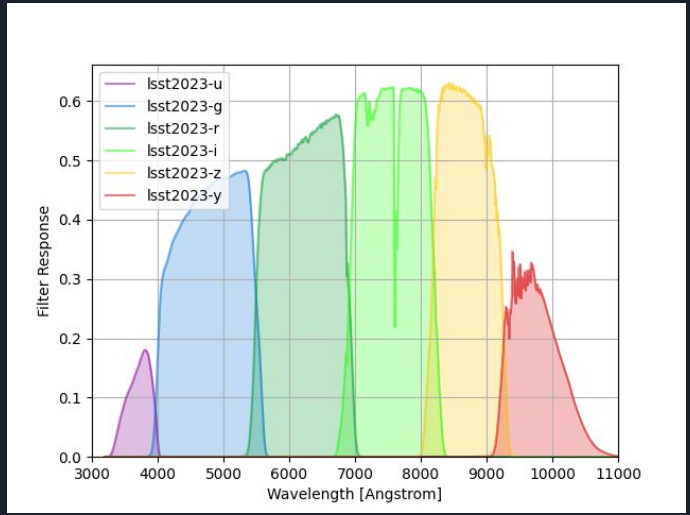
Data

Our data consists of the flux measurements of galaxies from images taken with 6 different filters as well as the angular resolution of the galaxy
(farther galaxies will be theoretically smaller)

Each filter is sensitive to a range of wavelengths within the ultraviolet through near infrared spectrum

The Astrolensing Group images galaxy clusters in 5–6 of these filters

Signal to noise is improved by combining images of the same filter together. Overall we achieve an exposure depth of 2000 seconds (30 minutes)

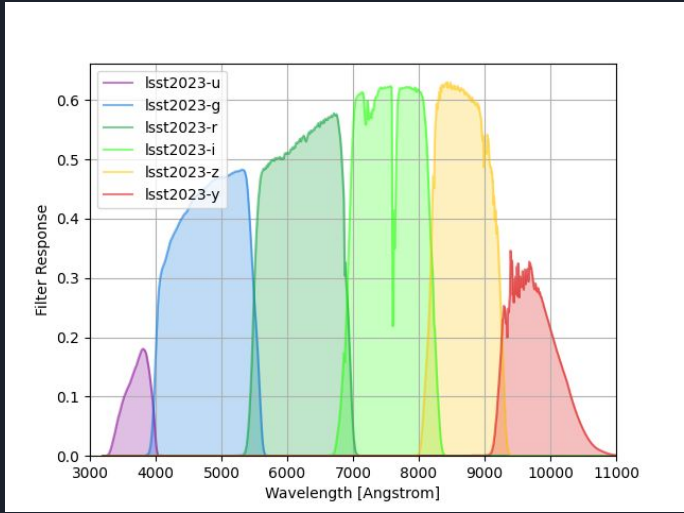


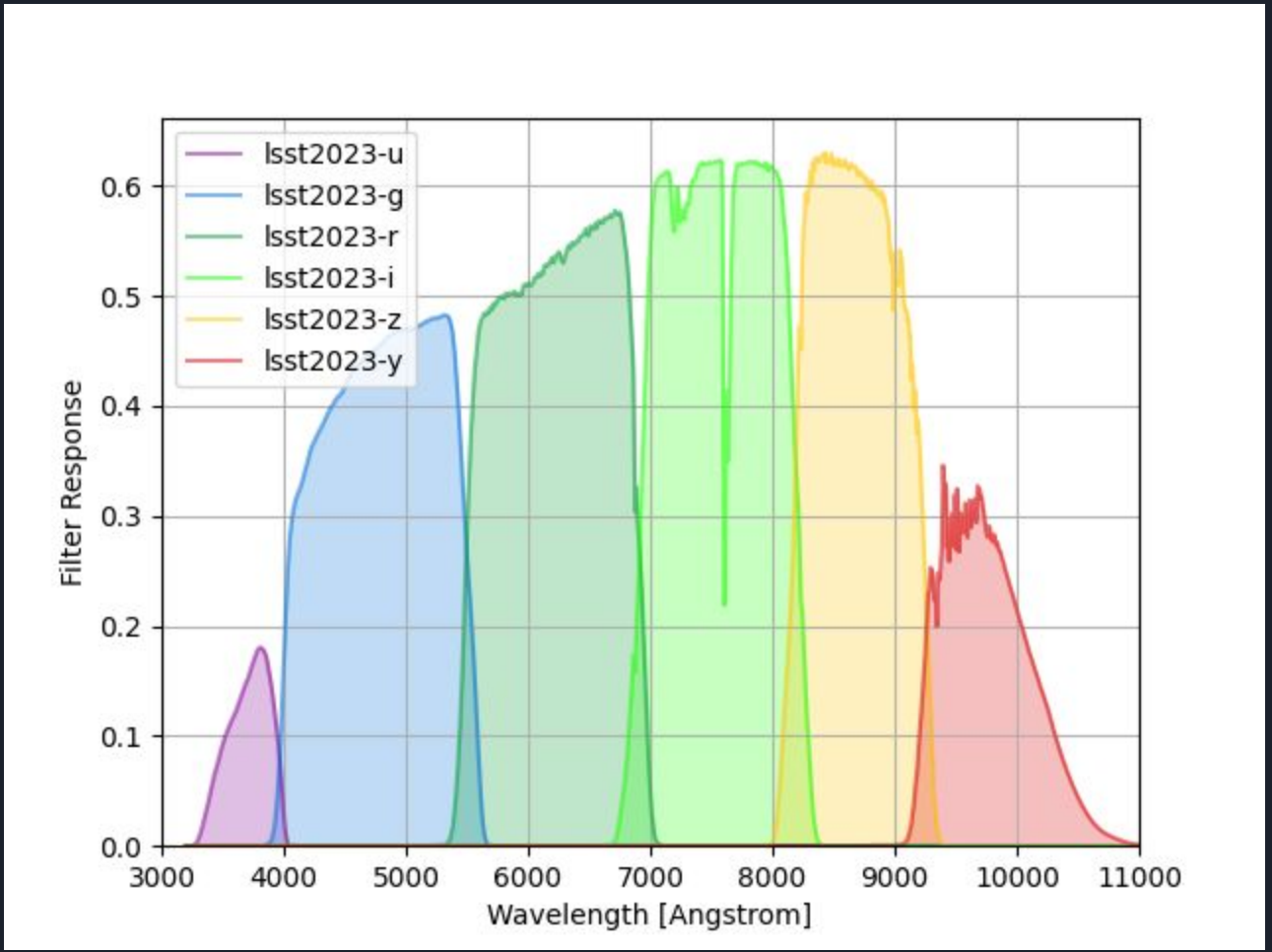
Data

Our pipeline locates the galaxies in the stacked images (versus stars) and extracts shape and flux information

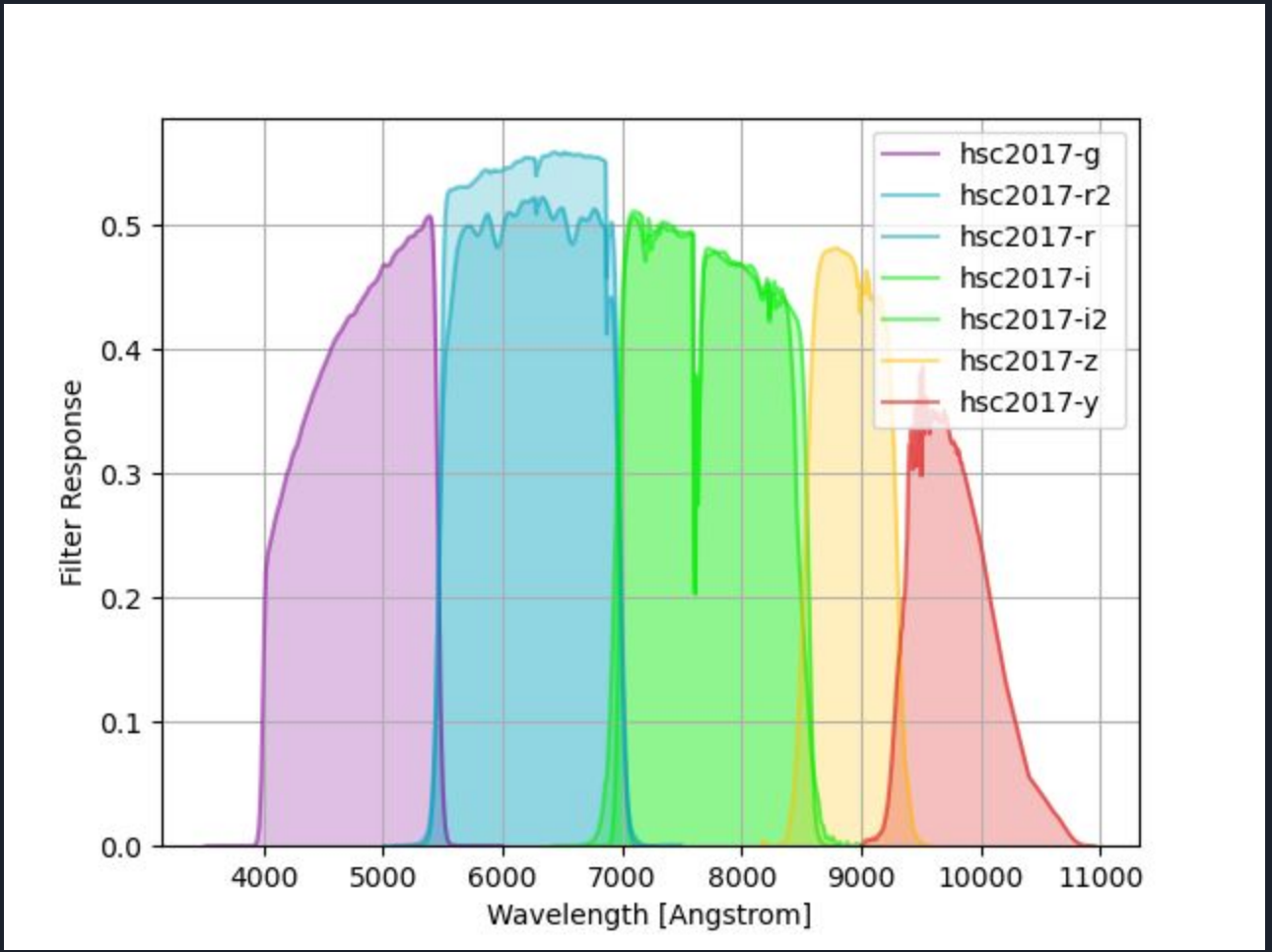
For this machine learning project we use these 5–6 flux values of a galaxy as our input and try to predict the redshift of the galaxy

Unfortunately, not all observatories use exactly the same filters → our model may not work for other observatories (See Hyper Suprime-Cam comparison)





https://speclite.readthedocs.io/en/latest/filters.html

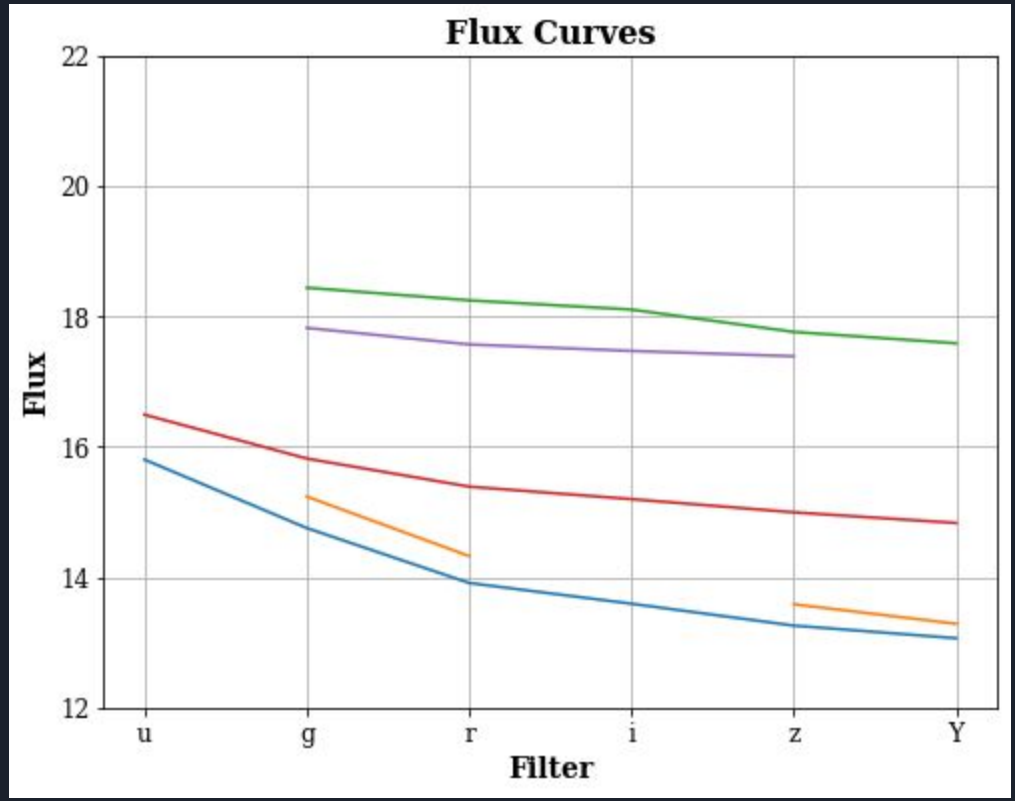


https://speclite.readthedocs.io/en/latest/filters.html

Data (Cleaning)

Some galaxies do not have enough S/N to give us an accurate flux, or the flux in that band was never measured at all.

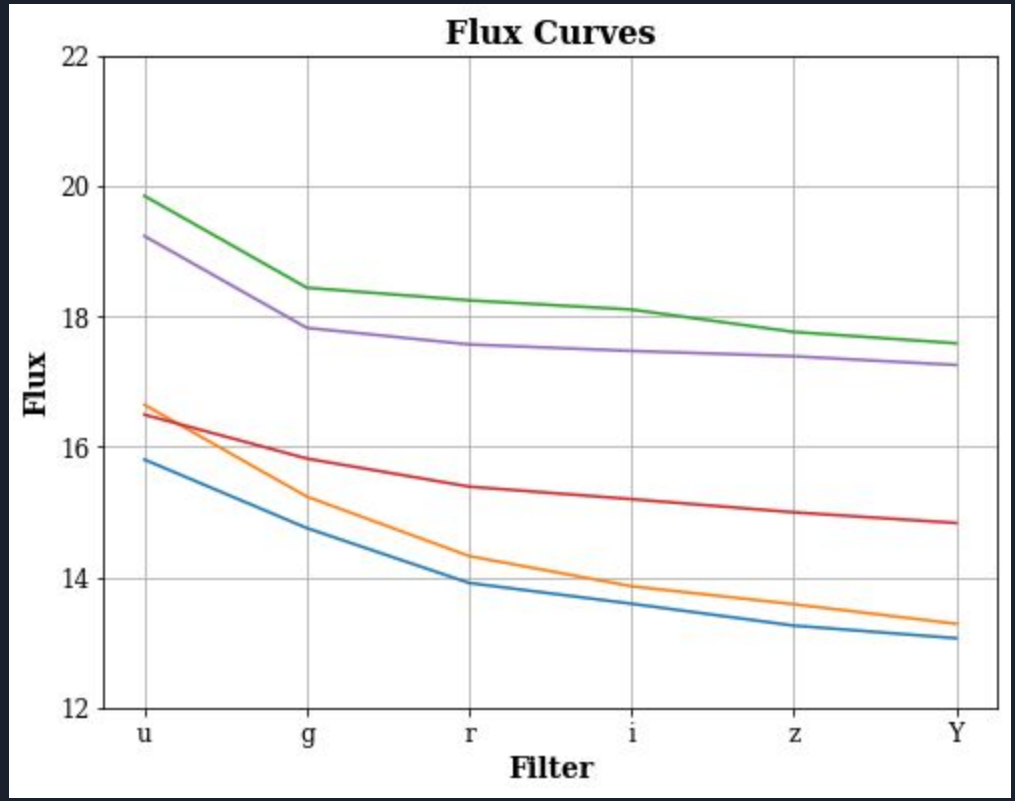
This leaves gaps in the flux curve of certain galaxies, and we need a way to fill those gaps in.



Data (Cleaning)

We can iterate through all the available data to find the average difference in magnitude between each band, where the data is complete.

Then we can infer the values of the missing fluxes
(from both sides if possible, averaging if two predictions are available)

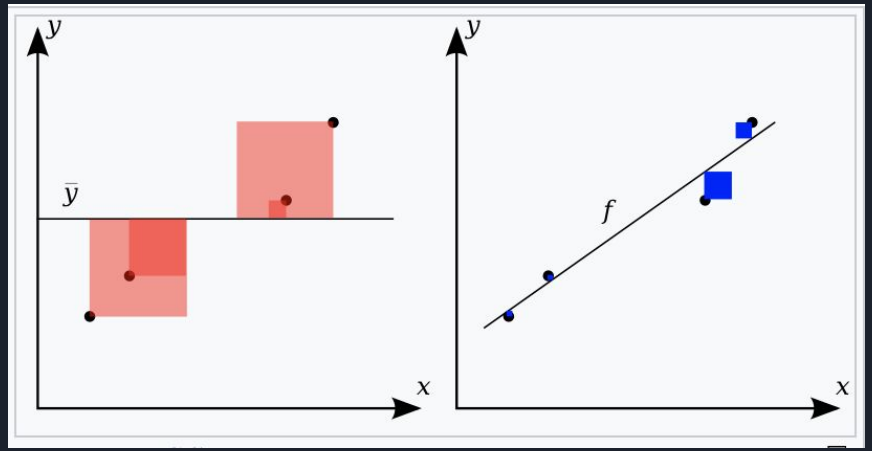


Accuracy Metric: R^2 coefficient of determination

Given residual sum of squares: $$SS_{\text{res}} = \sum_i (y_i - \hat{y}_i)^2$$

And total sum of squares: $$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$$

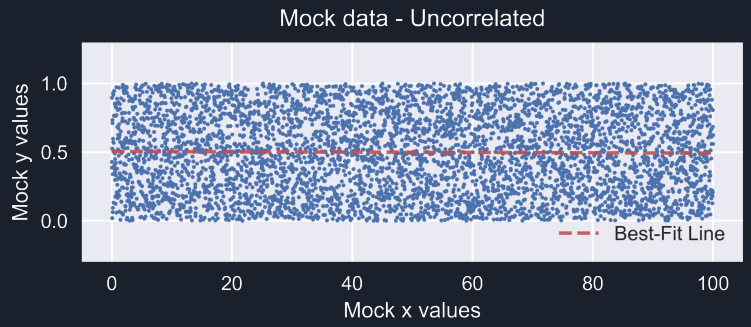
The R^2 coefficient is given by: $$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

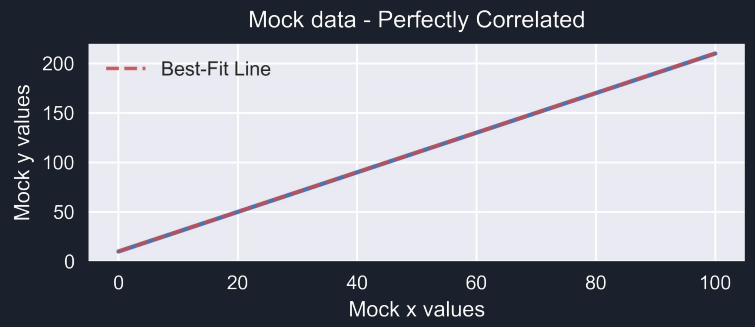


Accuracy Metric: R^2 coefficient of determination

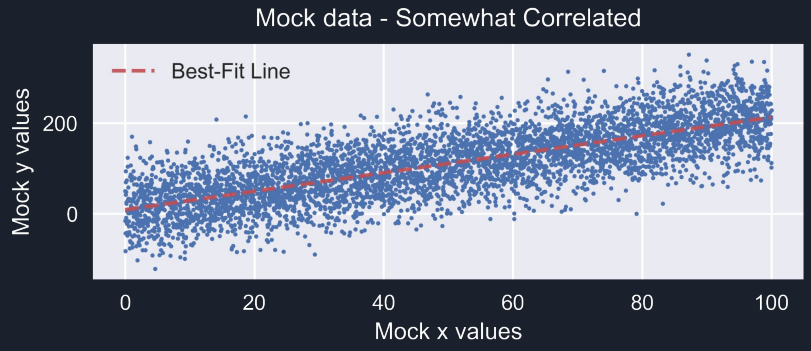
Range:
between 0 and 1

0: uncorrelated
1: perfectly correlated





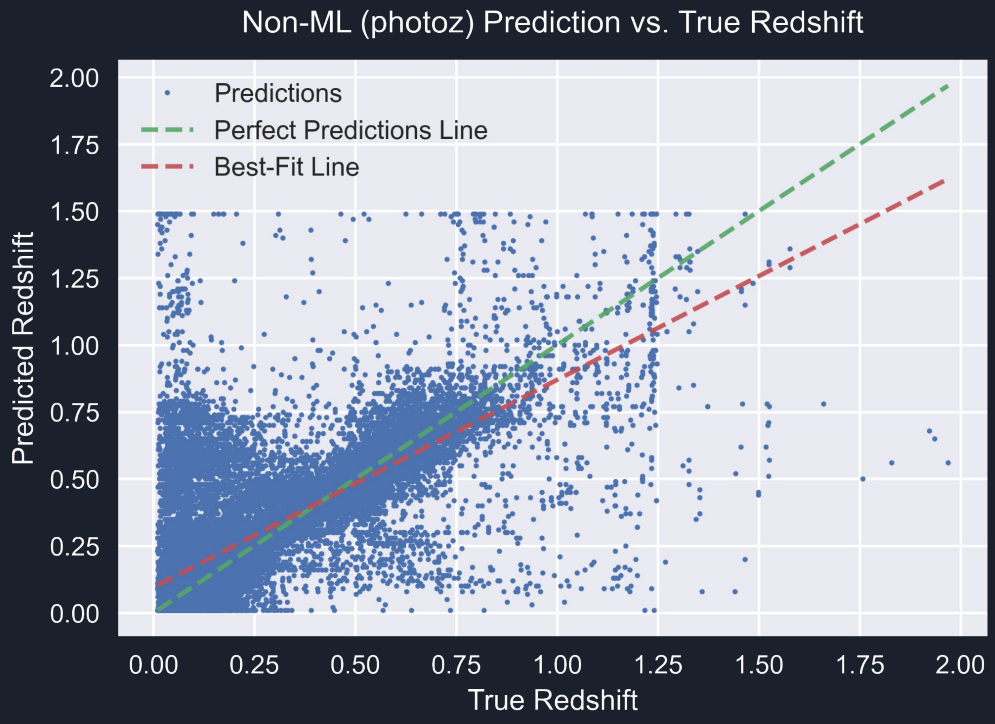
$R^2 = 0.58$



$R^2 = 0$

$R^2 = 1$

Non-ML (photoz) Method

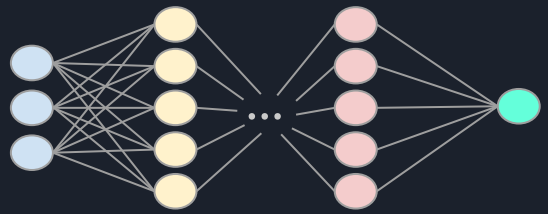


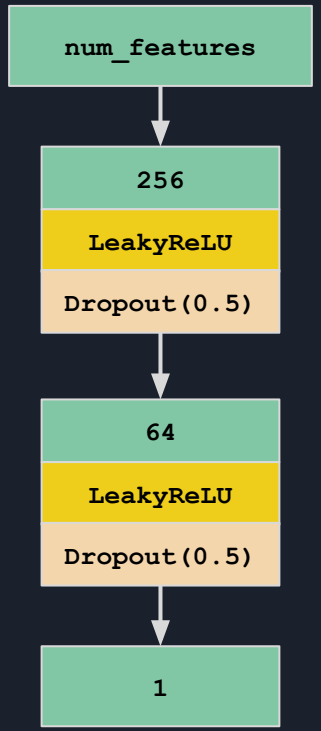
Traditional Non-ML method shown on the left

Mean-squared error: 0.0259

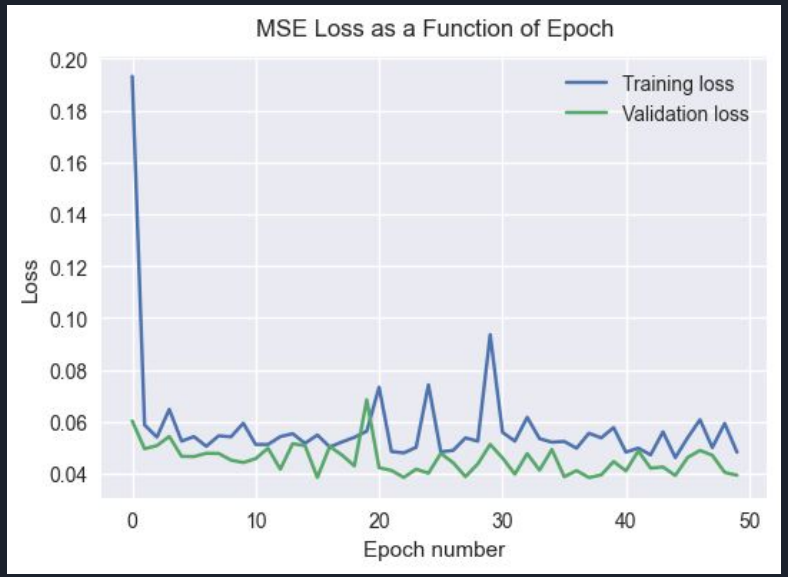
R^2 coefficient of determination: 0.594

Model 1: Fully-Connected NN First Version

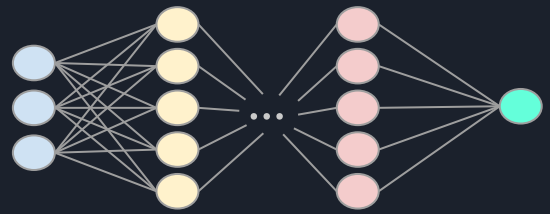


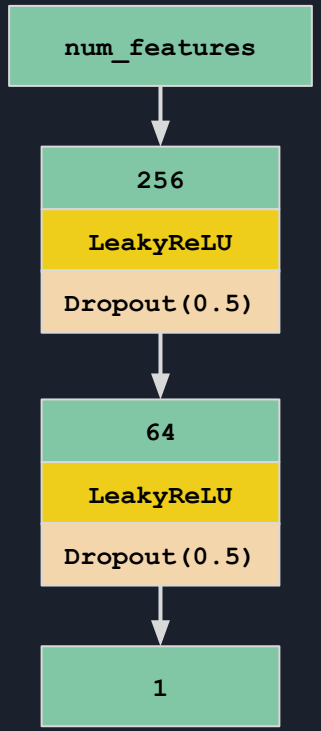


With 0.01 learning rate without normalization

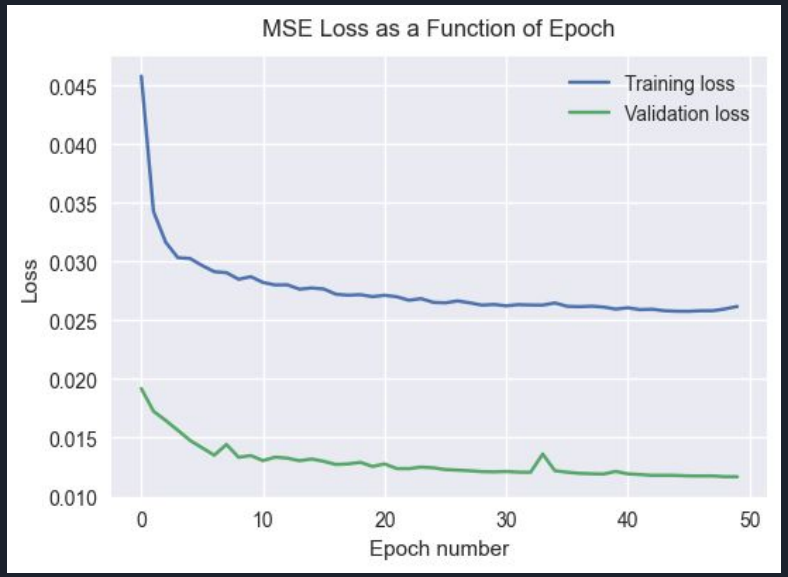


Model 1: Fully-Connected NN Second Version

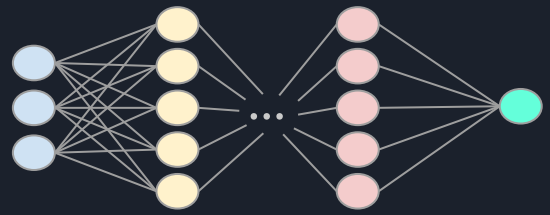


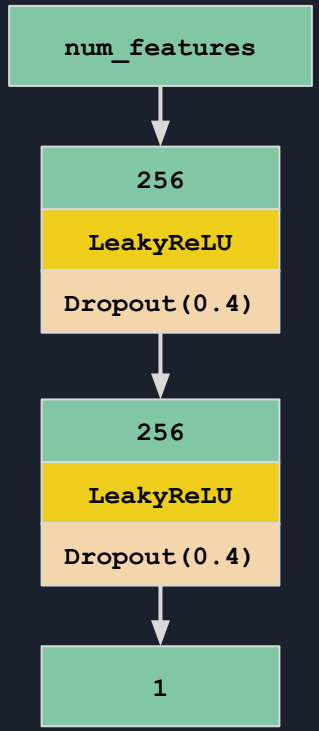


With 1e-4 learning rate and proper normalization

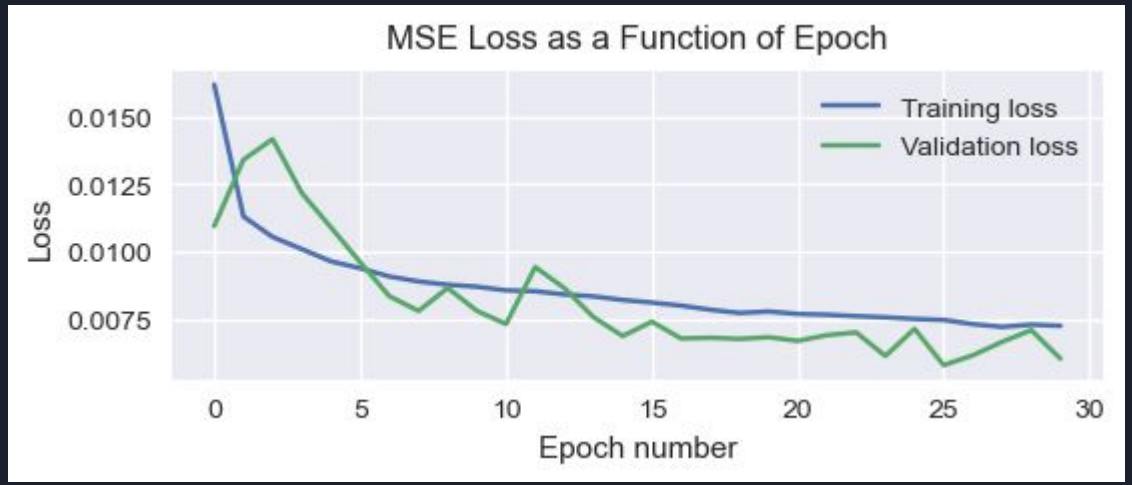


Model 1: Fully-Connected NN Final Version

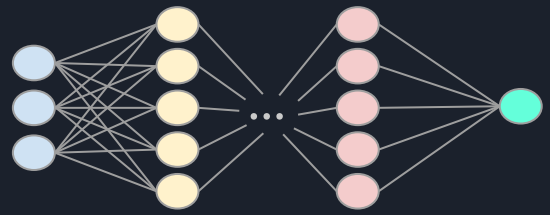


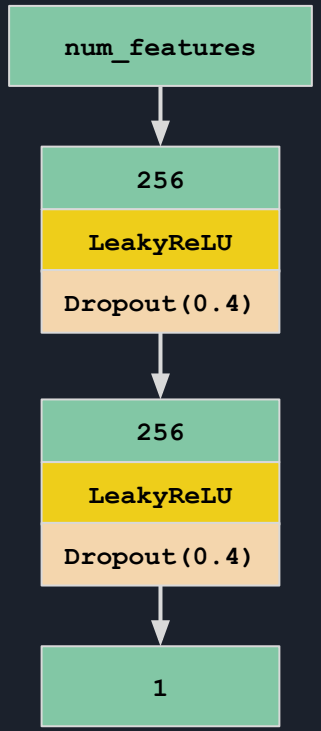


With 1e-4 learning rate with normalization



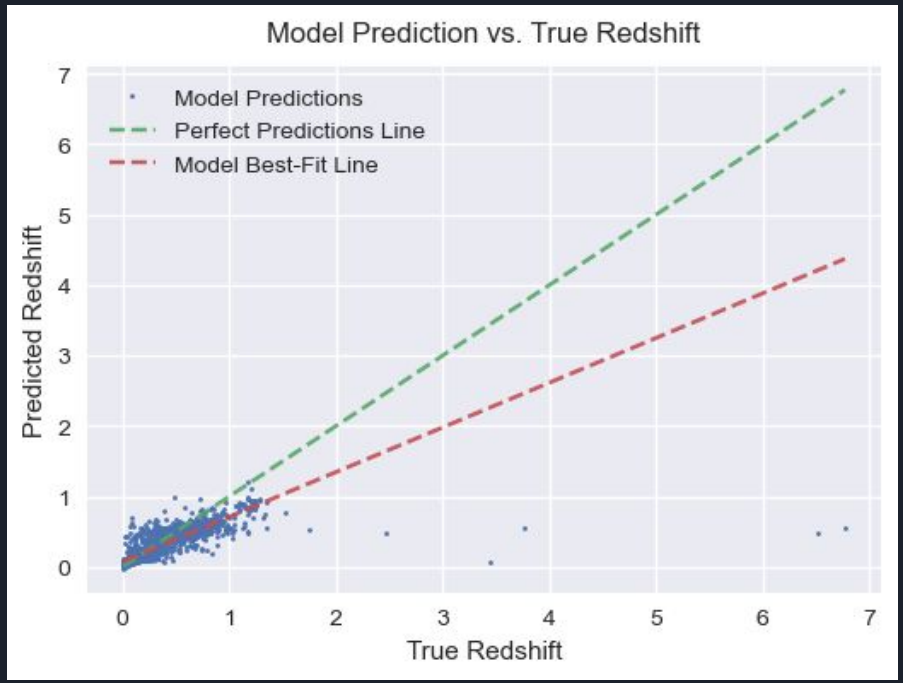
Model 1: Issue with Outliers



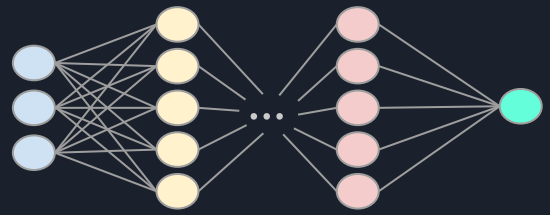


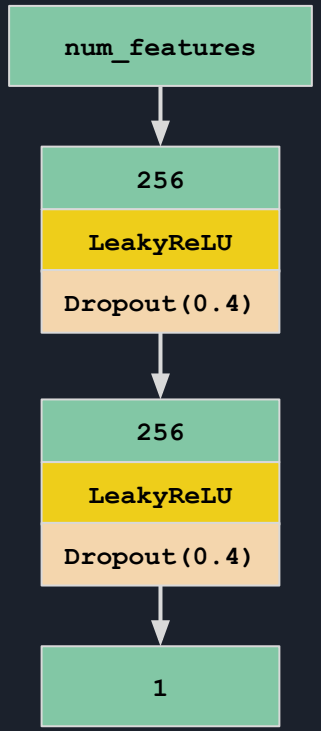
Poor performance on outliers

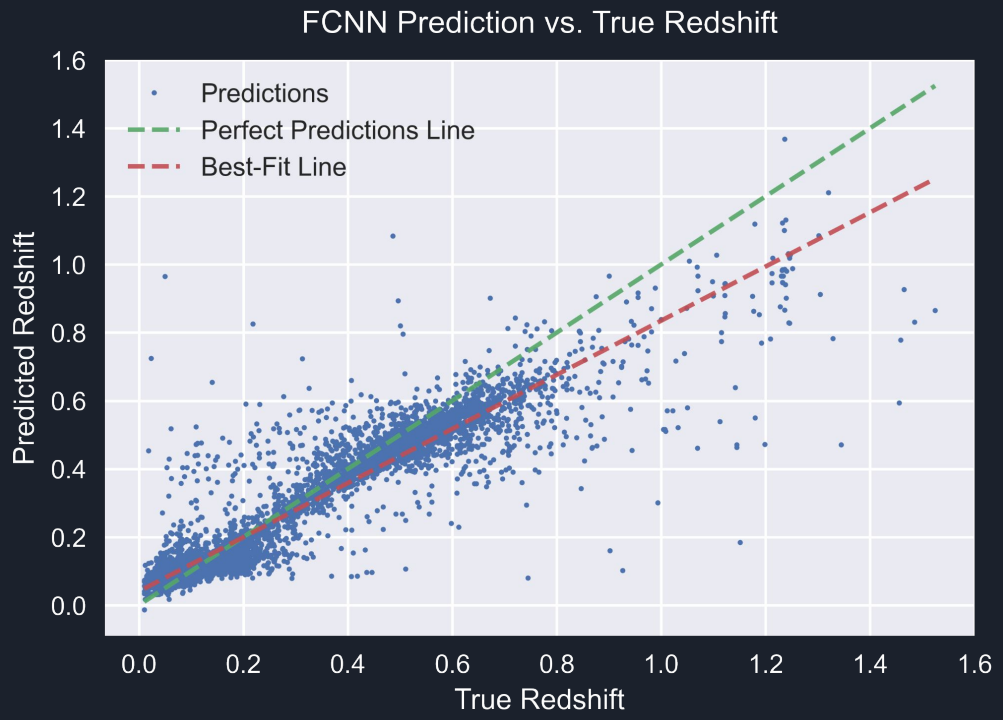
For the rest of our analysis, we implement a hardcoded redshift cutoff at z=2



Model 1: Fully-Connected NN





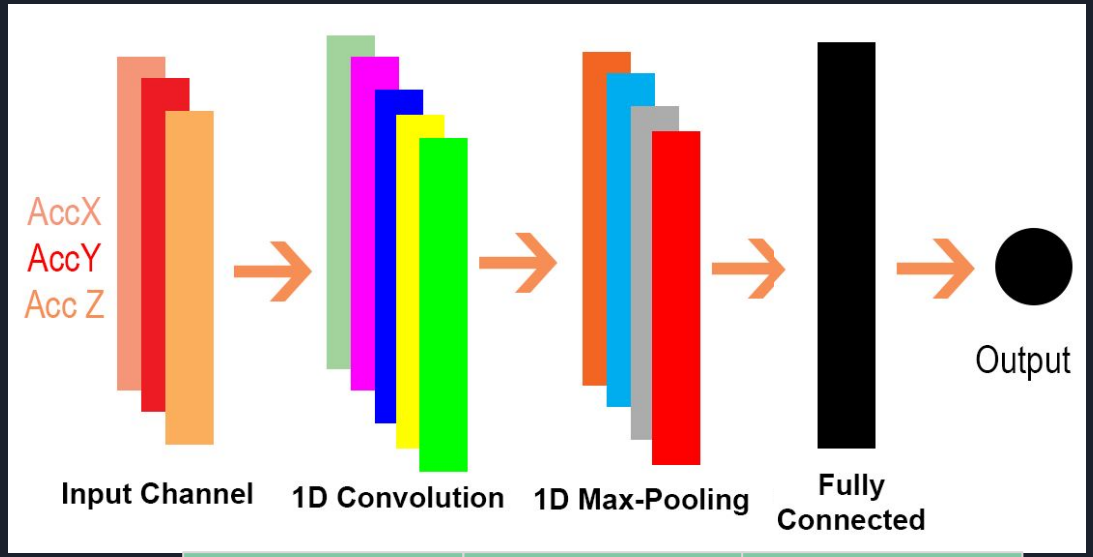


MSE on the test set: 0.0066

R^2 coefficient of determination: 0.893

Model 2: 1D CNN

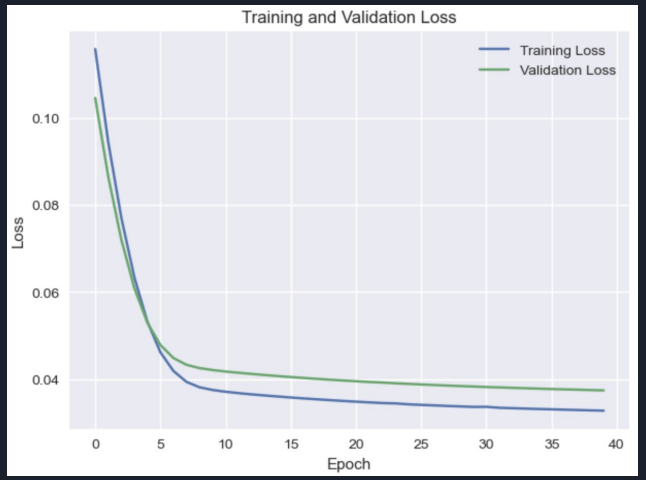
- Dataset as one-dimensional sequences of each galaxy cluster, with the magnitude measurements along one axis.



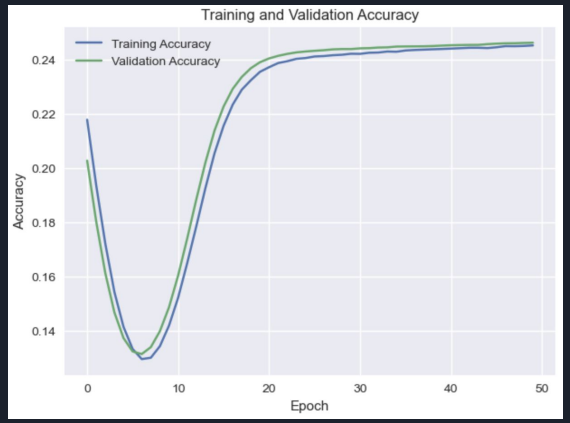
`in_channels=1`
`out_channels=16`
`kernel_size=3`
`stride=1`
`padding=1`
	`kernel_size=2,`
	`stride=2,`
	`padding=0`
		`Linear(16 *`
		`(num_features`
		`// 2),`
		`num_classes)`

Model 2: 1D CNN

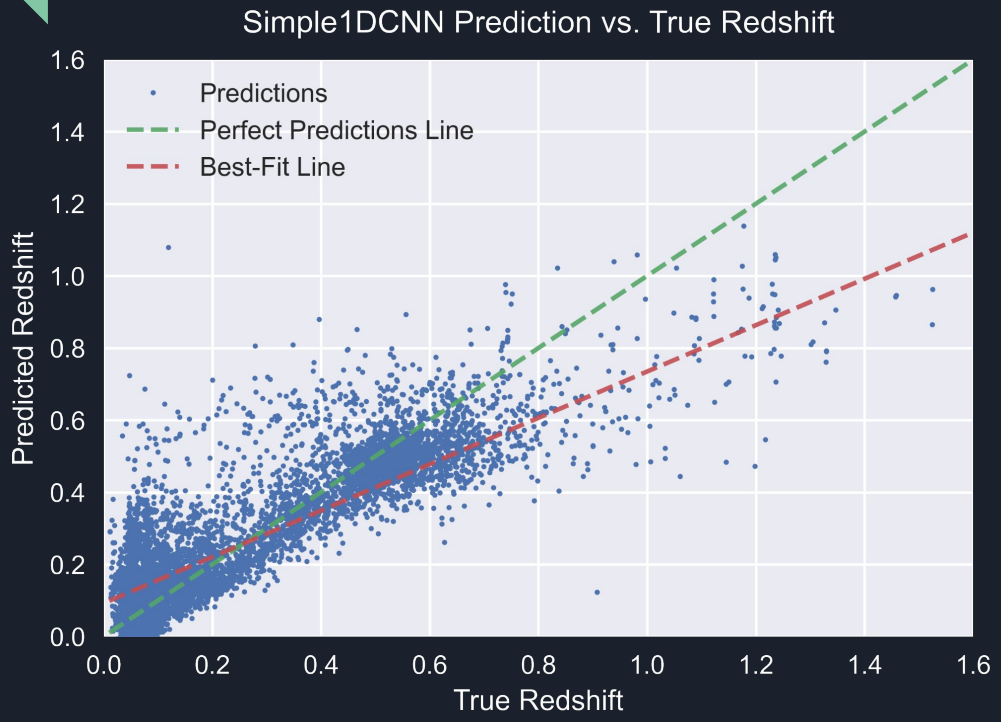
- Training and validation loss are close in values and decrease to about 0.03
- Epochs > 5 show only marginal improvements in loss reduction



- The MeanAbsoluteError accuracy reaches a plateau around 0.25 after 10 to 20 epochs
- Accuracy can decrease at first, likely due to data shuffling



Model 2: 1D CNN



Final learning rate: 1E-6
Optimizer: RMSprop
Batch size: 16

MSE on the test set: 0.0225

R^2 coefficient of determination: 0.655

Model 2: 1D CNN

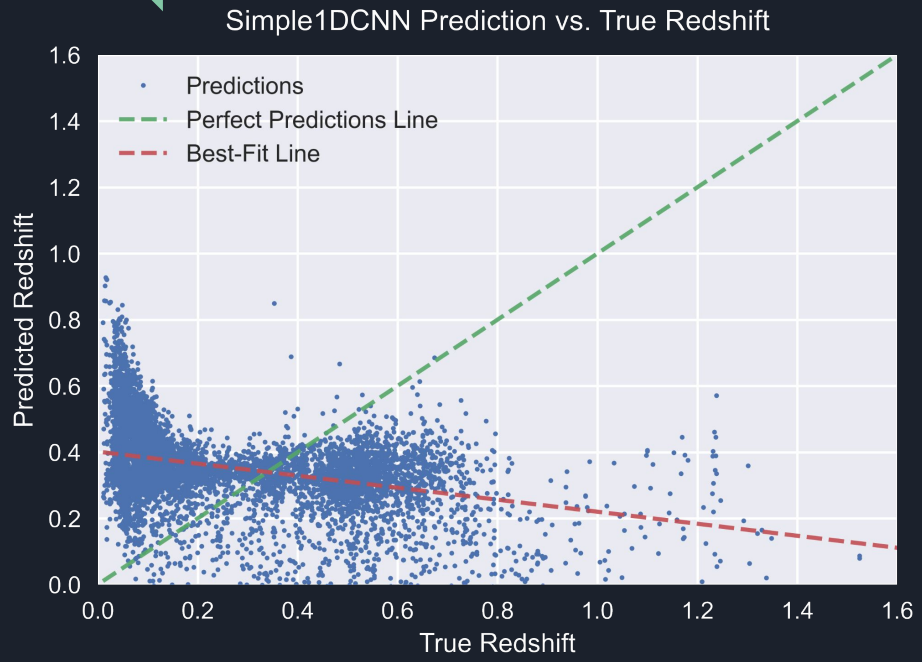
- Lower learning rate and smaller batch size generally improves performance

Optimizer	Learning rate	Batch size	Other hyperparameters
- Adam - SGD - RMSprop	1e-4 ~ 1e-6	16 ~ 32	- No. of layers - Dropout - Activation function

- RMSprop has slightly better performance than the others likely because it is less sensitive to learning rate

- Having activation function such as ReLU actually worsens performance

Model 2: 1D CNN

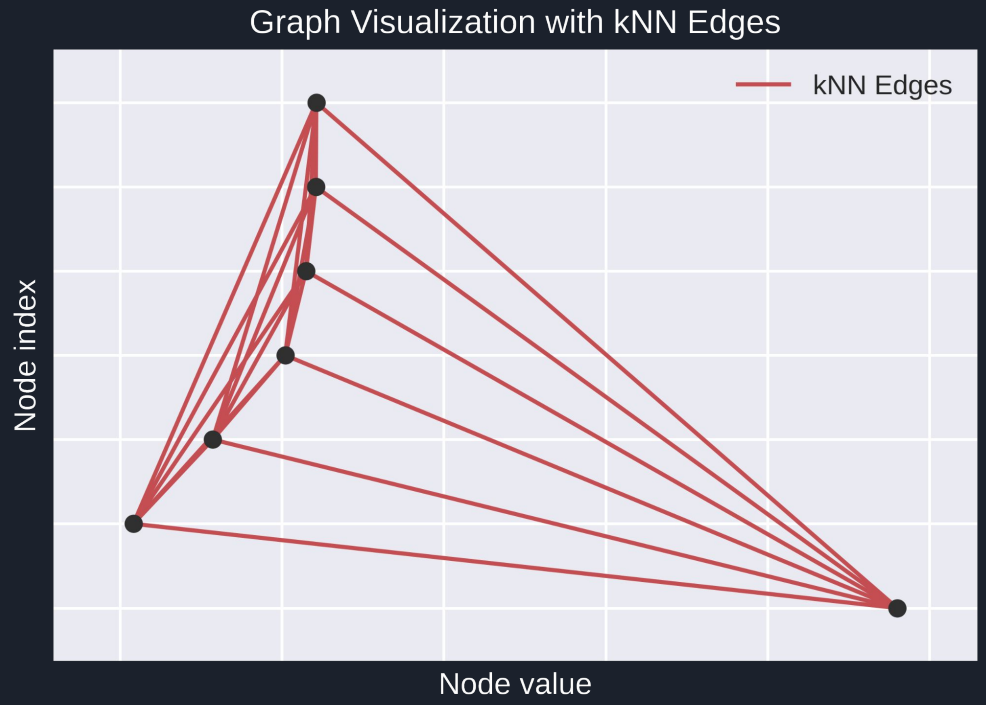


- There is a consistent drop in the 1D CNN model performance when non-linearity is introduced
- This means that the the relationship between the input features and the target variable might be predominantly linear
- 1D CNNs are designed to capture more complex patterns and spatial relationships within the data, which might not be necessary for simpler datasets

- Having activation function such as ReLU actually worsens performance

Model 3: Graph Attention Network

Review: How does it work?



In our data:

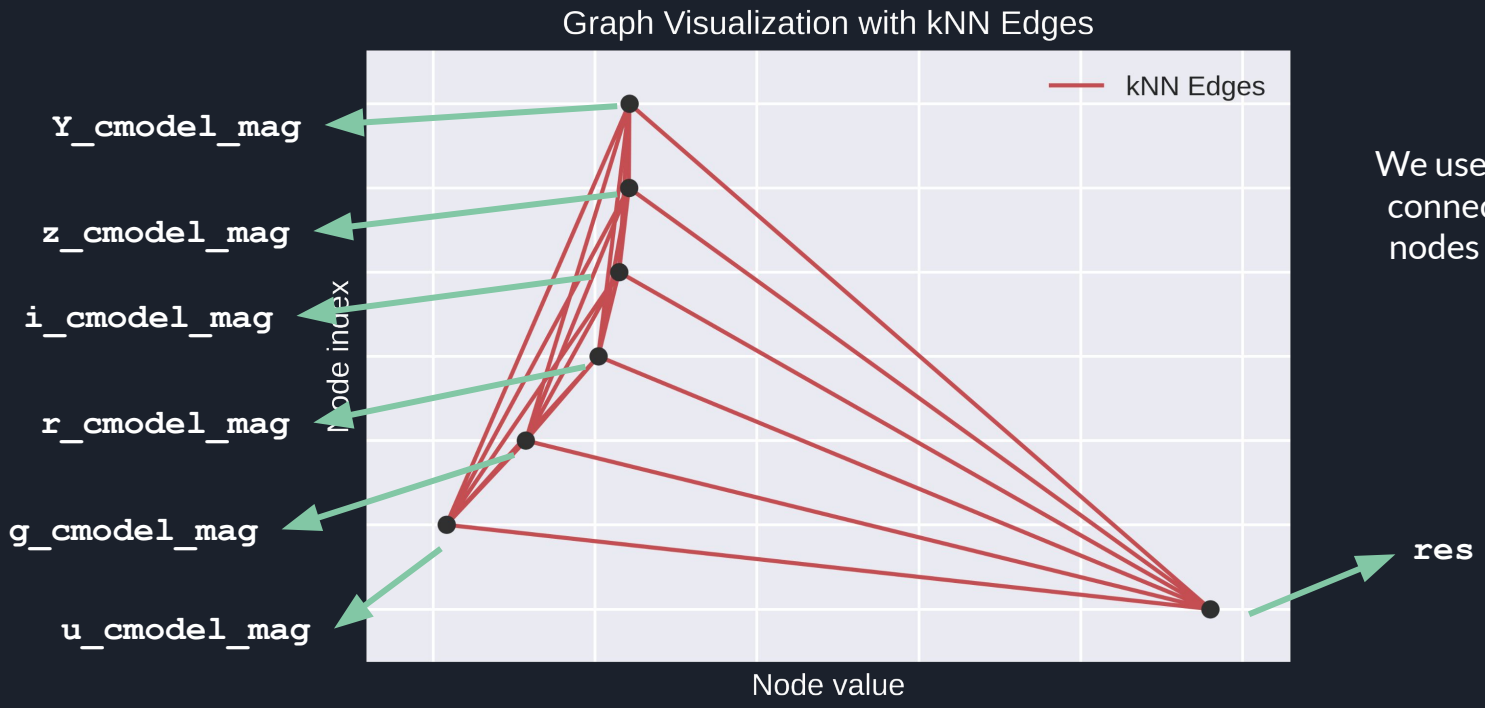
- ~50,000 galaxies
- Each galaxy has 7 features

Can construct 50,000 graphs with 7 nodes each

Each node has one feature (i.e. the value of that node)

Model 3: Graph Attention Network

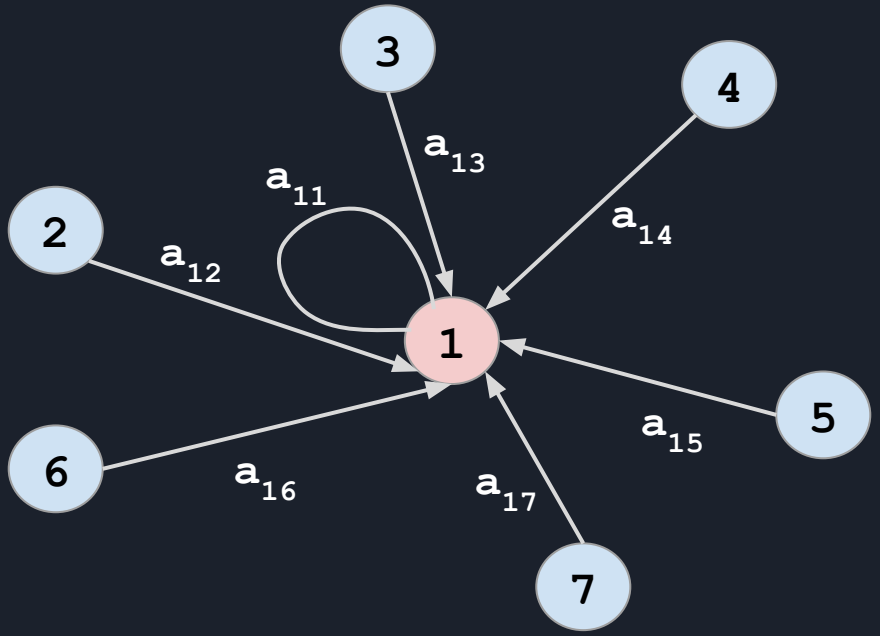
Review: How does it work?



We use 6 nearest neighbors to connect all nodes to all other nodes (using kNN algorithm)

Model 3: Graph Attention Network

Attention mechanism:



Focusing on node 1, how much attention should it pay to all the other nodes in the graph?

→ determined by learnable attention coefficients

How do we get these?

Model 3: Graph Attention Network

Attention coefficients: how do we get them?

A lot of possibilities, but let's focus on the method presented in Veličković et al. (2018)* since this is the default implementation in **`torch_geometric.nn.conv.GATConv`**

What about this?

$$e_{ij} = a\left(\vec{h}_i, \vec{h}_j\right) \longrightarrow \alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(e_{ik}\right)}$$

Compute unnormalized coefficients across pairs of nodes *i* and *j*

Apply softmax to obtain normalized weights

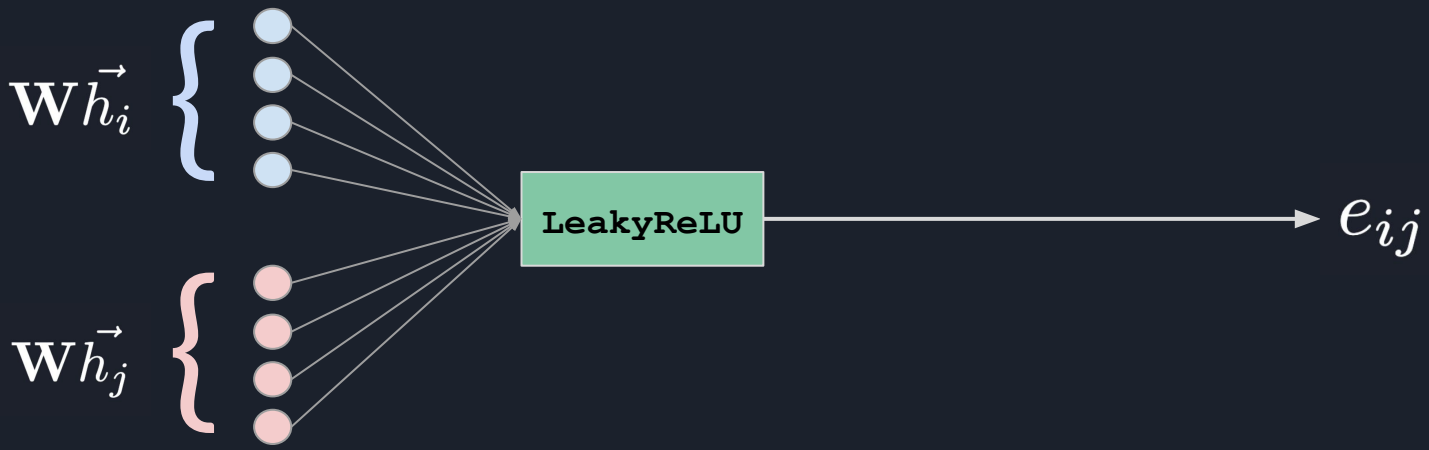
Graph Attention Networks, Veličković et al. (2018), arXiv:1710.10903

Model 3: Graph Attention Network

Attentional mechanism a

$$e_{ij} = a\left(\vec{h}_i, \vec{h}_j\right)$$

***W** is a matrix that maps the node features h to a higher dimension*



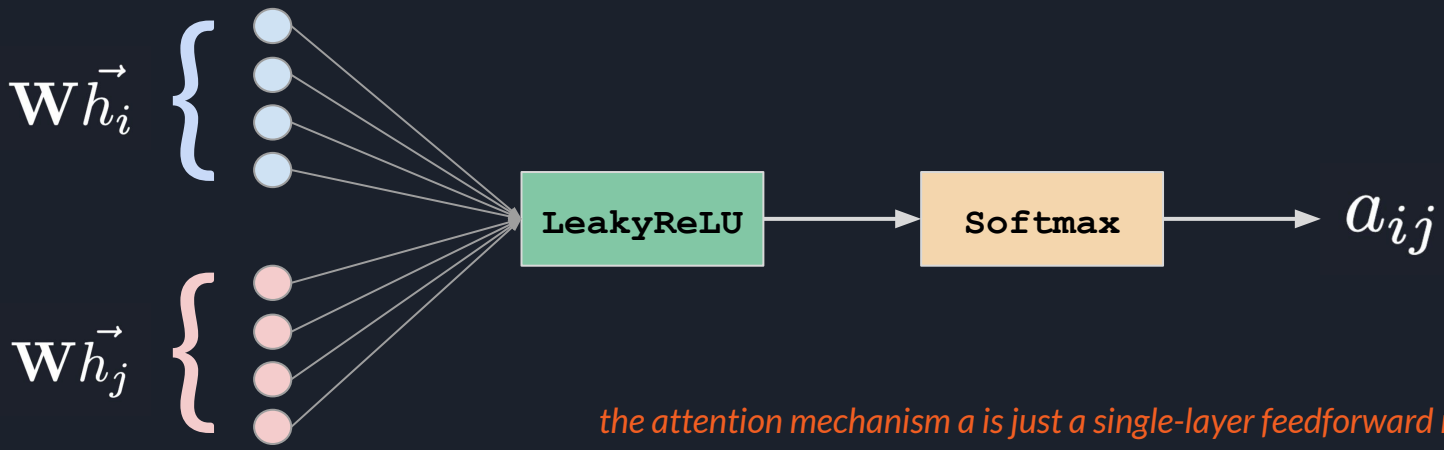
Graph Attention Networks, Veličković et al. (2018), arXiv:1710.10903

Model 3: Graph Attention Network

Attentional mechanism a

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(e_{ik}\right)}$$

***W** is a matrix that maps the node features h to a higher dimension*

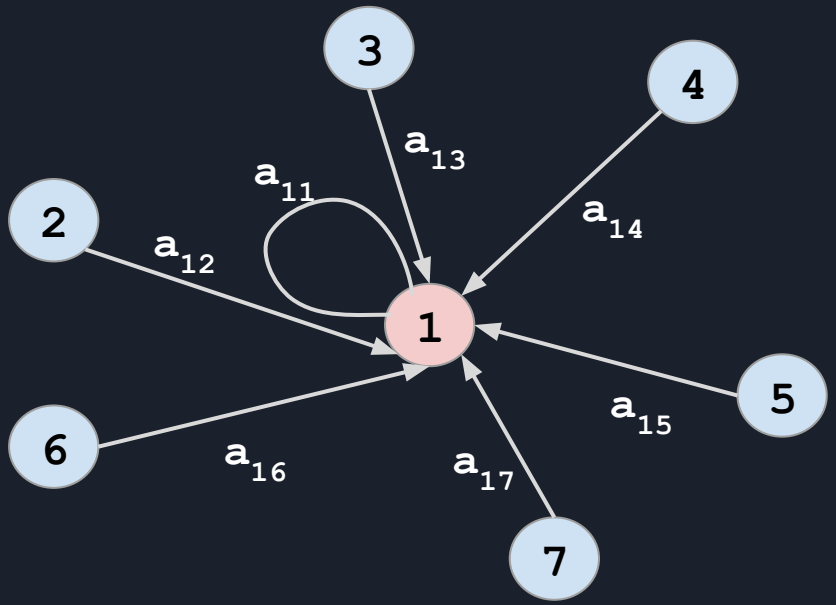


the attention mechanism a is just a single-layer feedforward neural network!

Graph Attention Networks, Veličković et al. (2018), arXiv:1710.10903

Model 3: Graph Attention Network

GAT Convolution:



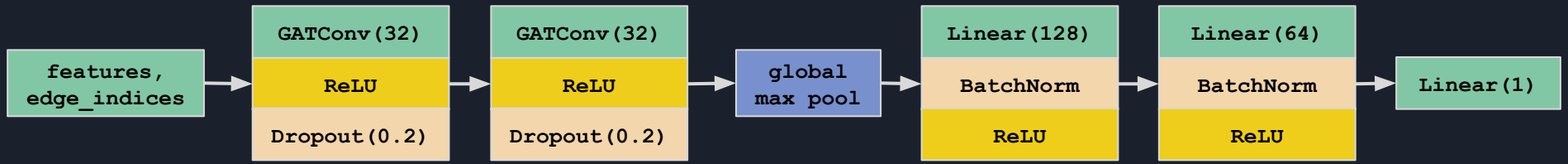
concat/avg

1′

Repeat for all nodes in the graph → one GATConv layer!

Model 3: Graph Attention Network

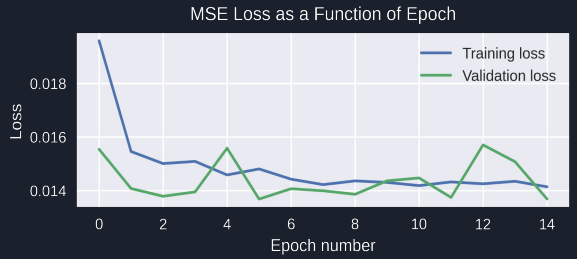
Architecture



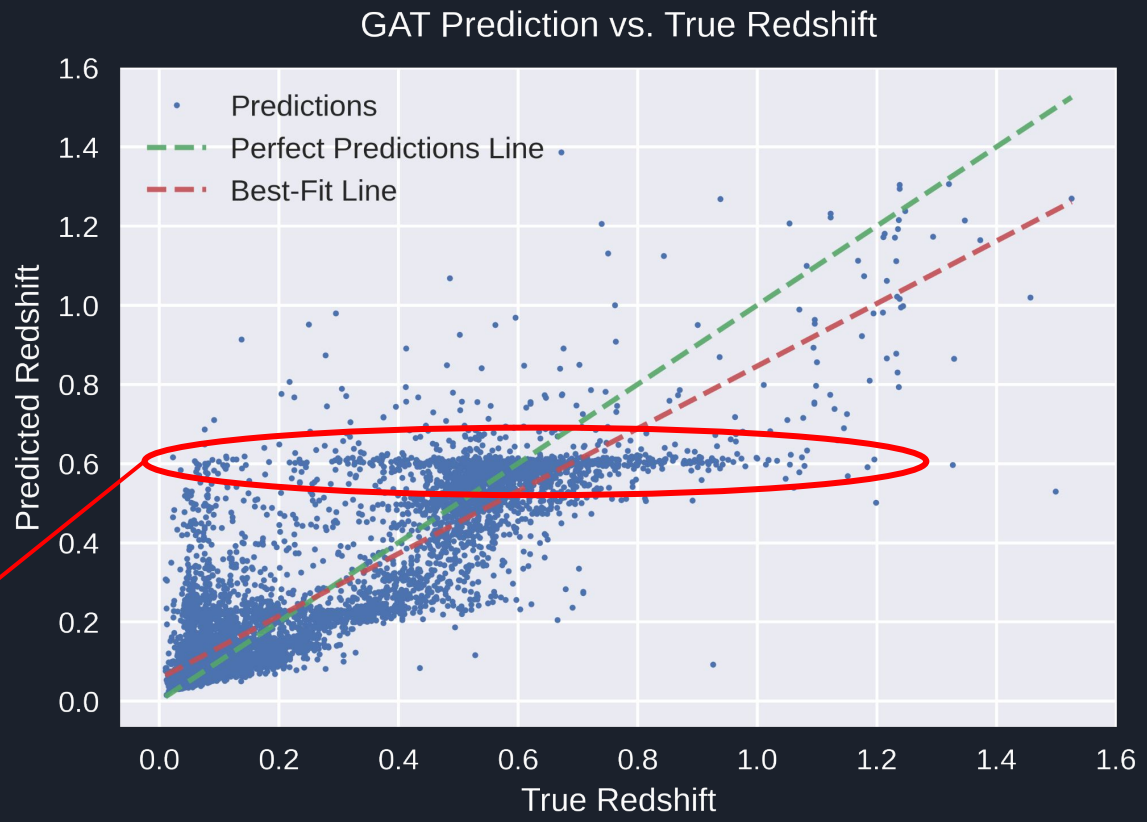
Learning rate:	1e-3
Optimizer:	AdamW
L2 Regularization:	Applied
Weight decay:	1e-5

Model 3: Graph Attention Network

Results: surprisingly poor performance

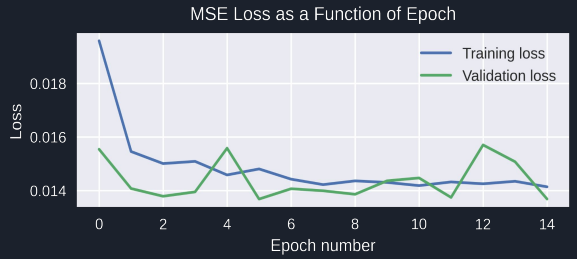


Weird artifacts showing up



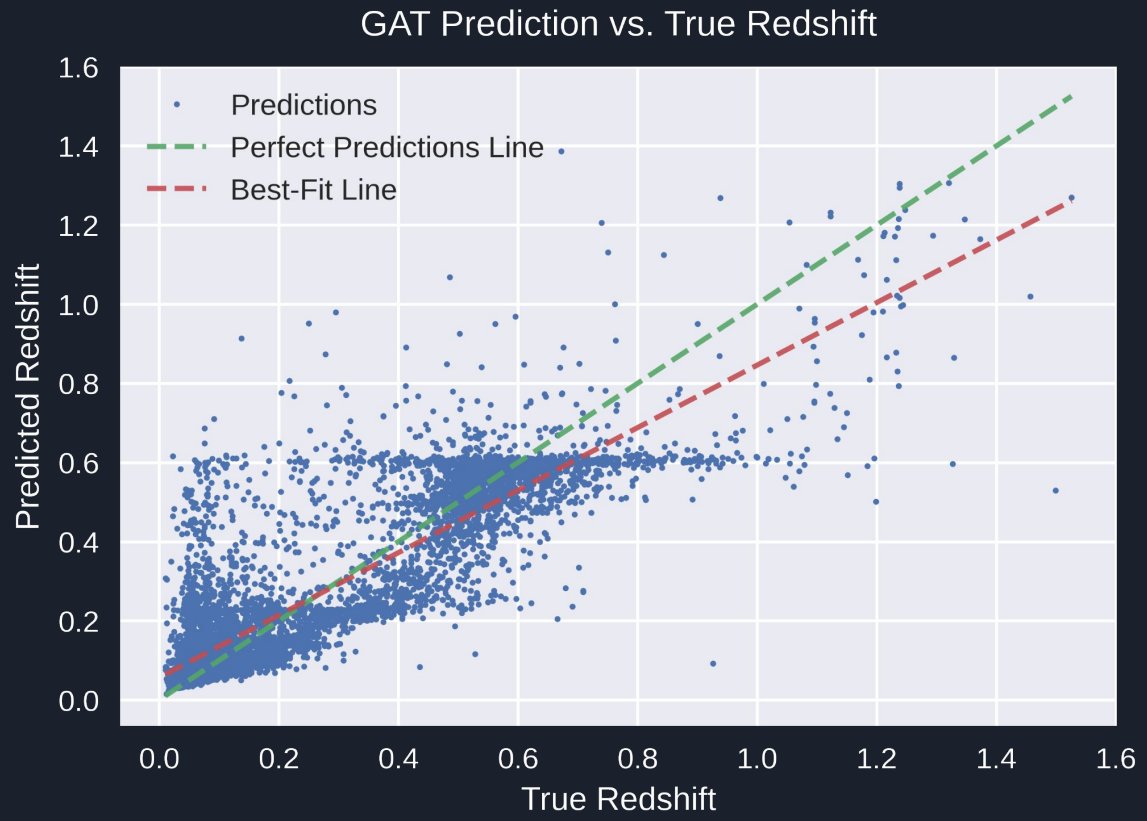
Model 3: Graph Attention Network

Results: surprisingly poor performance



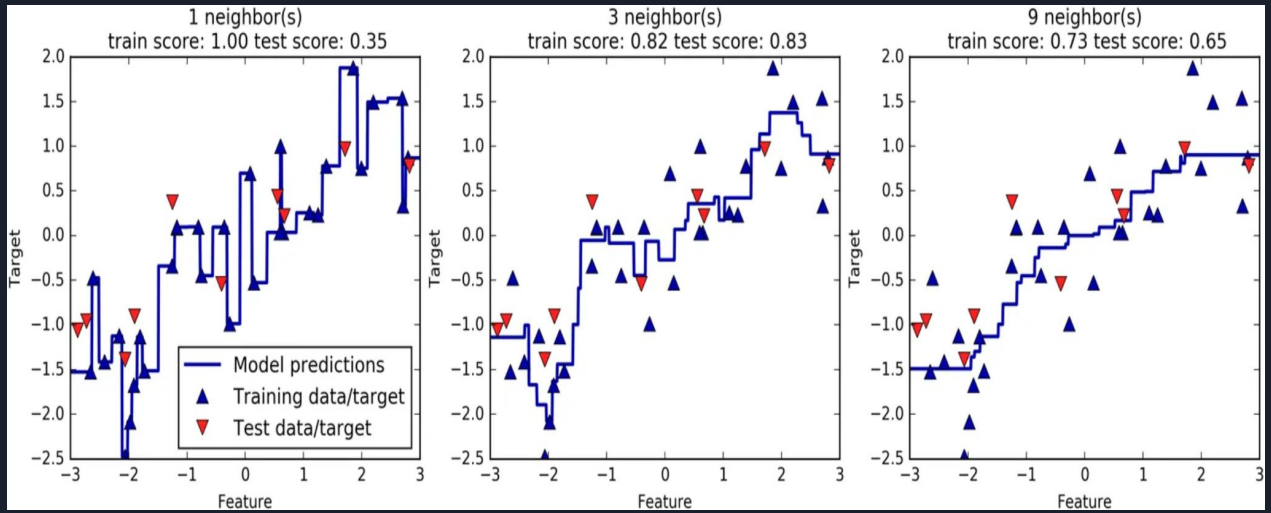
MSE on the test set:
0.0133

R^2 coefficient of determination:
0.759



Model 4: KNN Regression

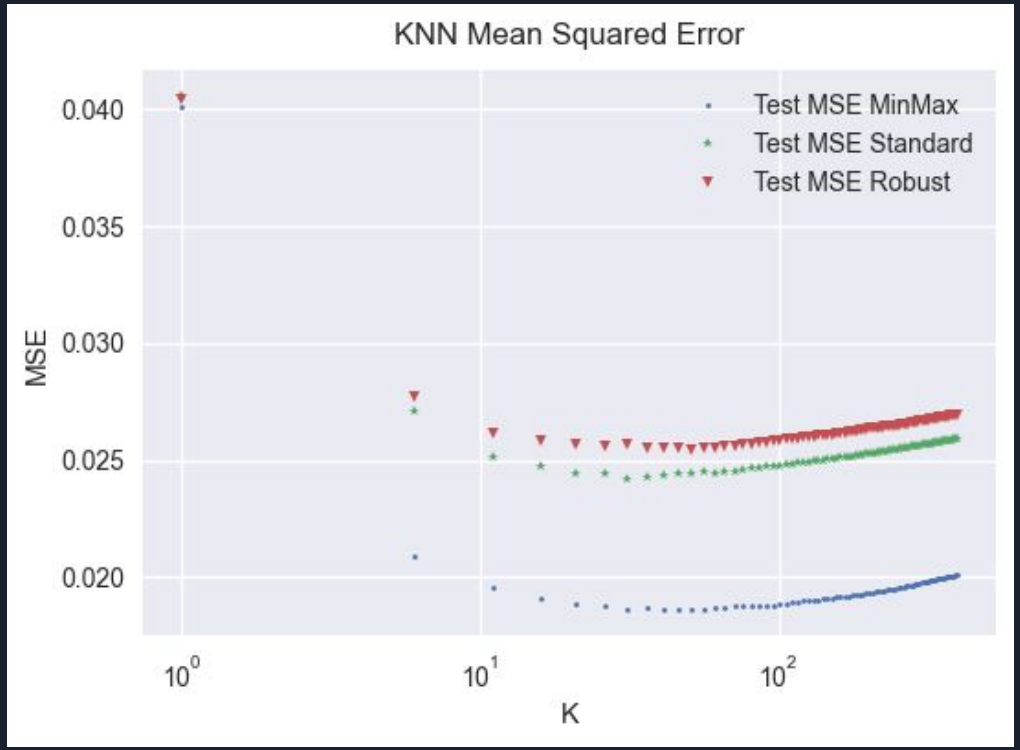
- Performs a regression analysis using k nearest neighbors
- 7-Dimensional feature space (res, u, g, r, i,z, Y)
- Euclidean distance metric for choosing nearest neighbors
- Arithmetic mean used for averaging



Credit: Mujajir, Imam, Medium

Model 4: KNN Regression

- Rescaling the data
- MSE metric for accuracy
- Can give varying weights to each neighbor

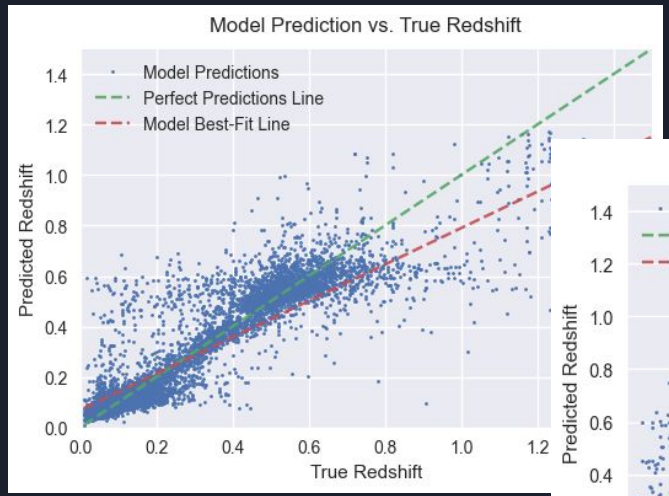


Optimal K → MinMax: 51 Standard: 31 Robust: 51

Model 4: KNN Regression

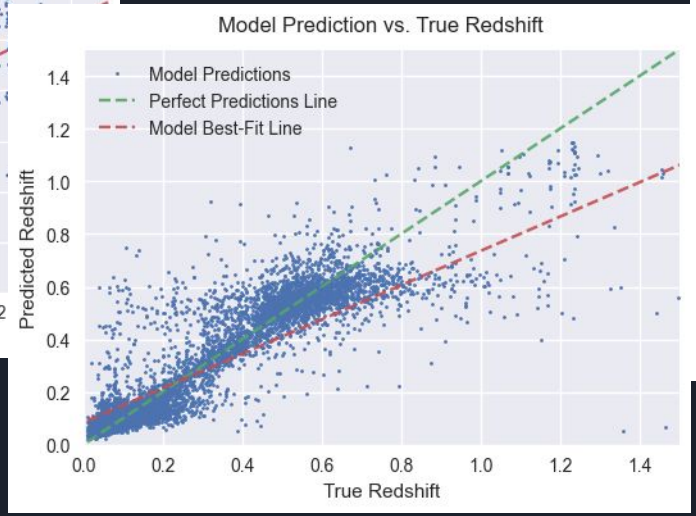
Minmax scaling

MSE: ~0.019



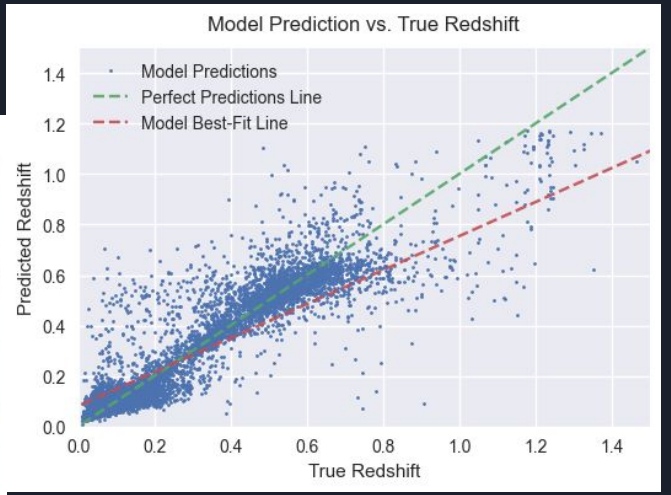
Robust scaling

MSE: ~0.025



Minmax scaling

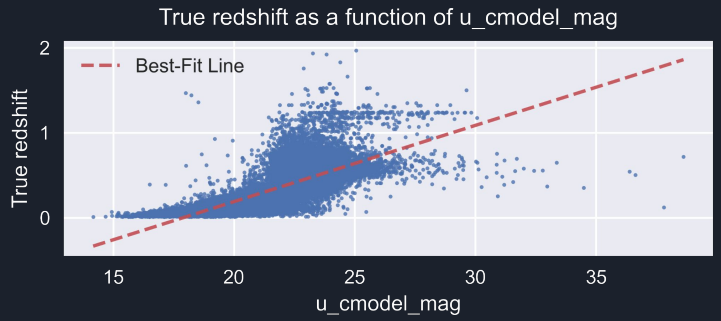
MSE: ~0.024



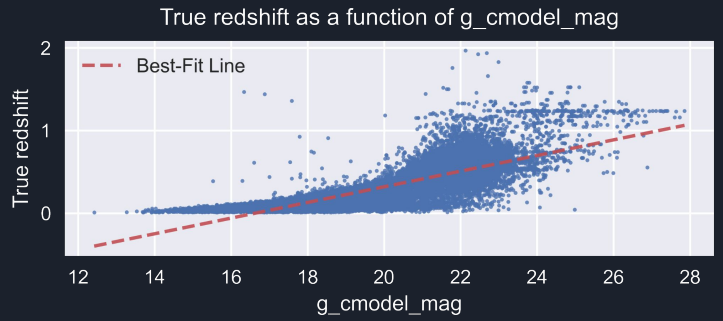
Results Summary

Model	FCNN	1D CNN	GAT	KNN Reg.	Non-ML (photoz)
MSE on test set	0.0066	0.0225	0.0133	~0.019	0.0259
R^2 coefficient	0.893	0.655	0.759	~0.8	0.594

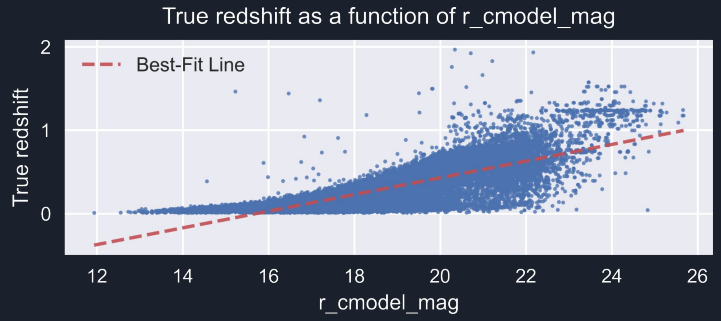
Why does the FCNN work?



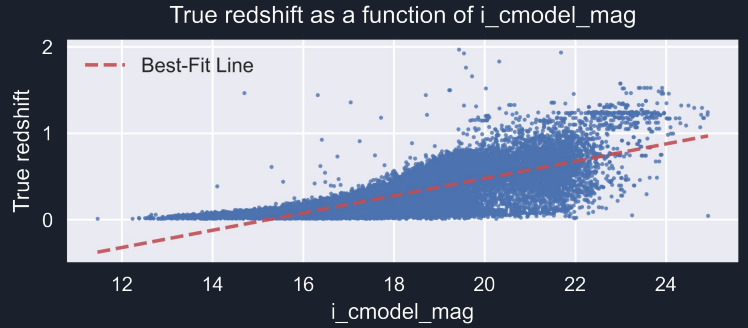
$R^2 = 0.70$



$R^2 = 0.75$



$R^2 = 0.69$



$R^2 = 0.63$

Discussion

- FCNN outperforms all other models → data is not complex enough for more advanced architectures

- Convolution layers may not be compatible with the data, as the data is relatively low-dimensional and may lack spatial or sequential dependencies

- A more careful analysis of data is needed before conclusions can be drawn

Next Steps

- Include images of galaxies themselves and perform 2D CNN?
- Test the model on data from other observatories
- Make the model more rigorous. Tune hyperparameters to values that make scientific sense, rather than just being a black-box

This precursor analysis could eventually improve the performance of our mass calculations down the line