Estimating galaxy shape and flux with CNNs

Final project CS109b Spring 2020

Ziwei Qui, Hayden Joy, Zach Murray, and Dan Cox

Problem Statement:

We have been given a file of 18779 simulated galaxy images and asked to use them to:

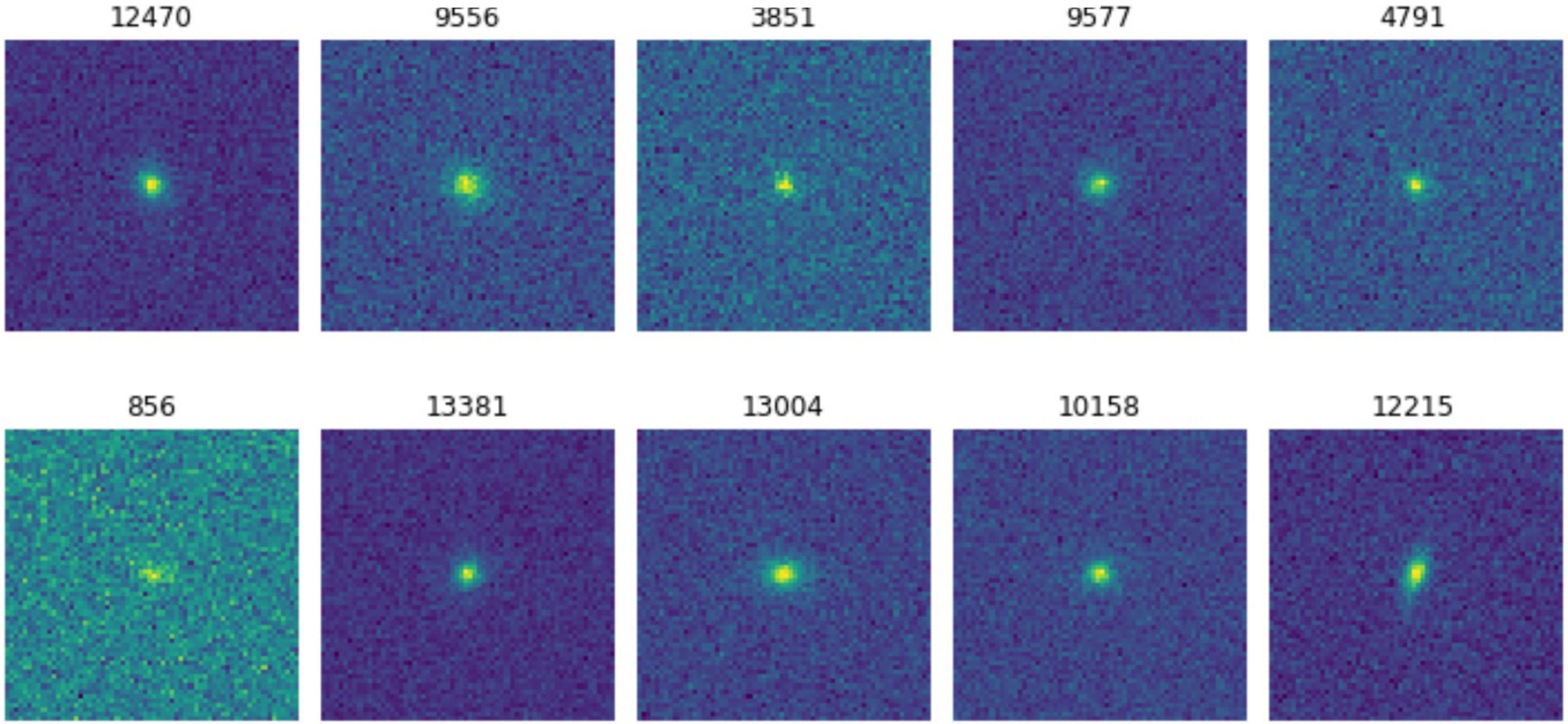
1. Train a CNN to estimate five parameters commonly associated with galaxy images.
2. Assess the CNN's performance on a sample of mock images.

Motivation:

Such a tool could be used to describe and then classify new galaxy's as they are identified.

A sample of the Galaxy Data

Ten random images from the dataset



The five parameters to be estimated

1. Flux — brightness divided by area
2. Sérsic index — the degree of curvature of the galaxy profile.
3. Sérsic radius — half-light radius
4. g1 — orientation
5. g2 — ellipticity

What we have done:

We have generated:

- 5 CNNs each estimating a single parameter.
- 1 CNN that estimates all parameters at once.

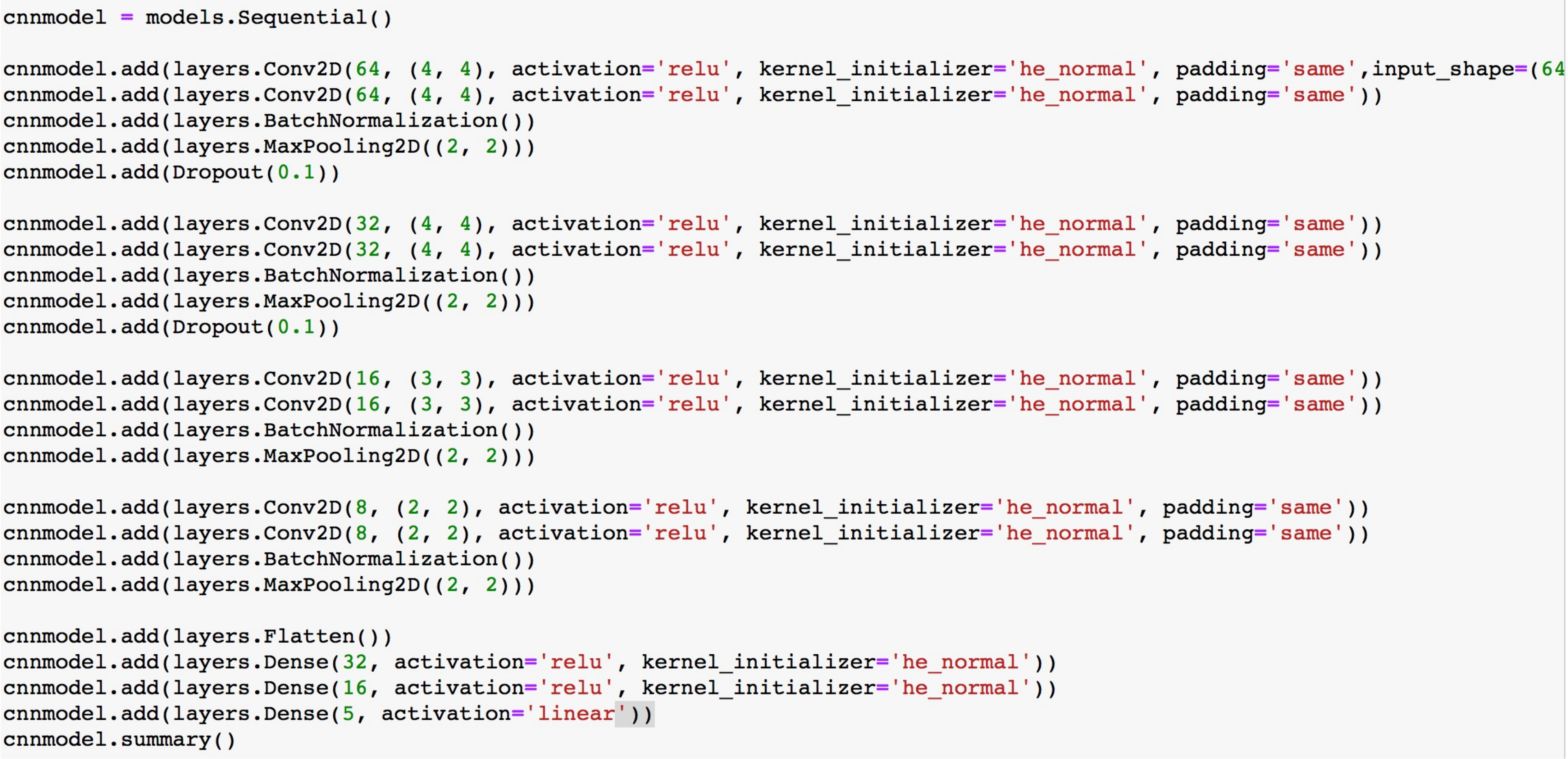
We have examined the performance of the models’ with respect to:

- Variations in background noise vis-à-vis the Cramer-Rao bound.
- Differences between the point spread function used in the training vs the testing data.
- The galaxies not being centered in the image

We have also done some baseline modeling not involving neural networks,
but rather more conventional approaches

1 CNN (5 outputs):

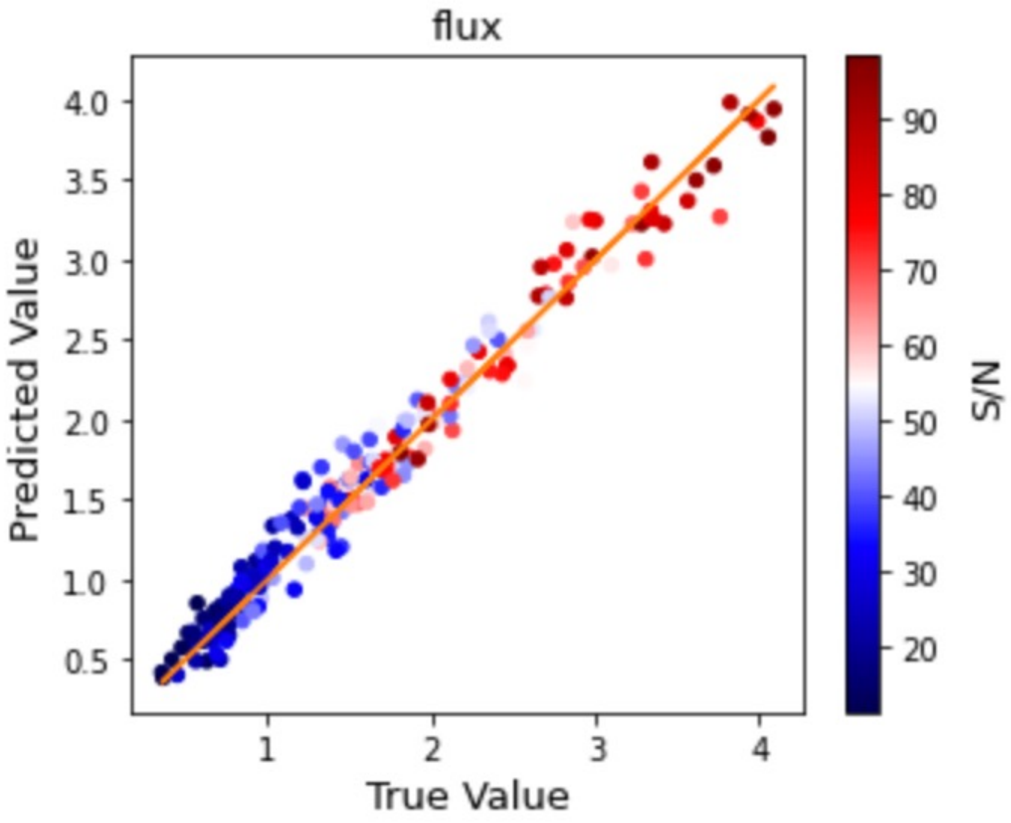
Architecture of the CNN (5 outputs):



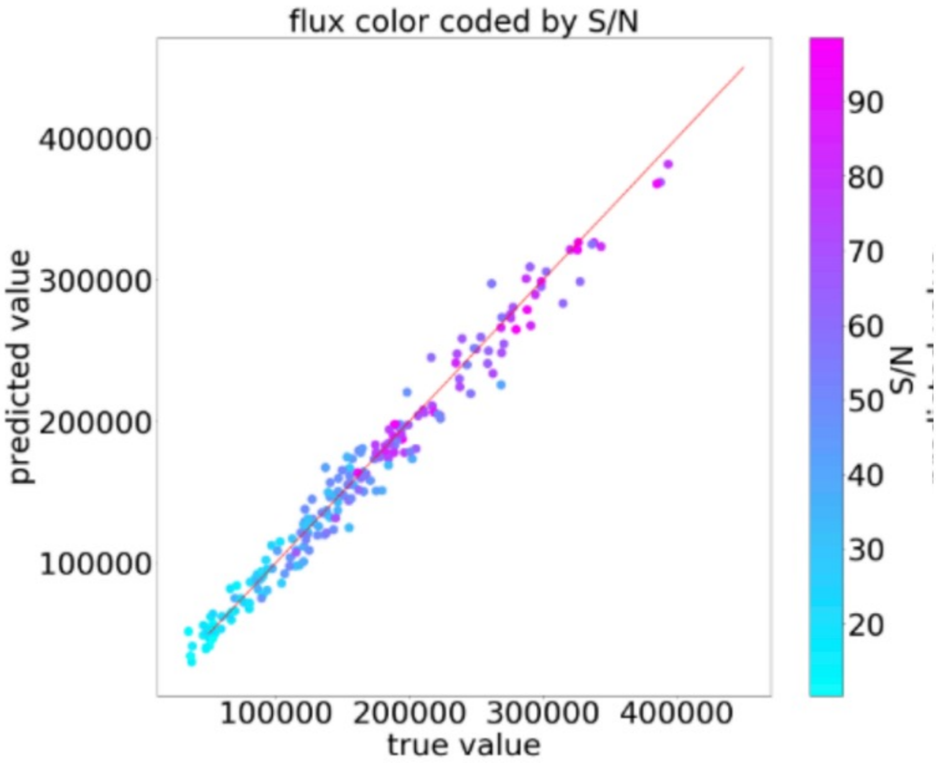
```
cnnmodel = models.Sequential()

cnnmodel.add(layers.Conv2D(64, (4, 4), activation='relu', kernel_initializer='he_normal', padding='same',input_shape=(64
cnnmodel.add(layers.Conv2D(64, (4, 4), activation='relu', kernel_initializer='he_normal', padding='same'))
cnnmodel.add(layers.BatchNormalization())
cnnmodel.add(layers.MaxPooling2D((2, 2)))
cnnmodel.add(Dropout(0.1))

cnnmodel.add(layers.Conv2D(32, (4, 4), activation='relu', kernel_initializer='he_normal', padding='same'))
cnnmodel.add(layers.Conv2D(32, (4, 4), activation='relu', kernel_initializer='he_normal', padding='same'))
cnnmodel.add(layers.BatchNormalization())
cnnmodel.add(layers.MaxPooling2D((2, 2)))
cnnmodel.add(Dropout(0.1))

cnnmodel.add(layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same'))
cnnmodel.add(layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same'))
cnnmodel.add(layers.BatchNormalization())
cnnmodel.add(layers.MaxPooling2D((2, 2)))

cnnmodel.add(layers.Conv2D(8, (2, 2), activation='relu', kernel_initializer='he_normal', padding='same'))
cnnmodel.add(layers.Conv2D(8, (2, 2), activation='relu', kernel_initializer='he_normal', padding='same'))
cnnmodel.add(layers.BatchNormalization())
cnnmodel.add(layers.MaxPooling2D((2, 2)))

cnnmodel.add(layers.Flatten())
cnnmodel.add(layers.Dense(32, activation='relu', kernel_initializer='he_normal'))
cnnmodel.add(layers.Dense(16, activation='relu', kernel_initializer='he_normal'))
cnnmodel.add(layers.Dense(5, activation='linear'))
cnnmodel.summary()
```

Results of the CNN (5 outputs):

Ours


flux

Benchmark

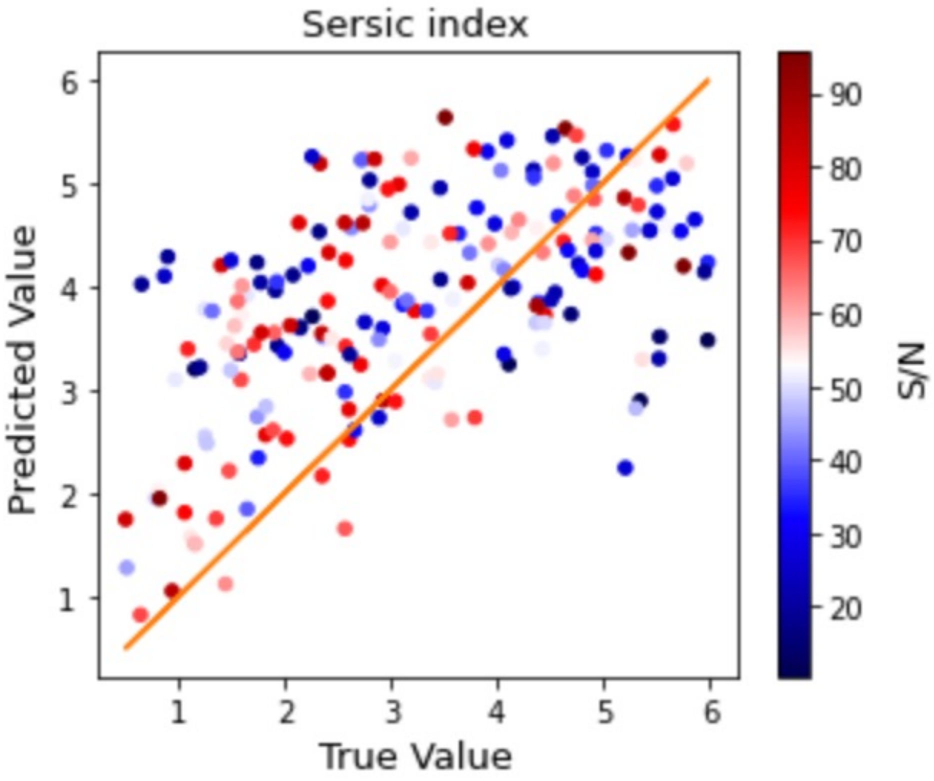

flux color coded by S/N

flux —brightness divided by area

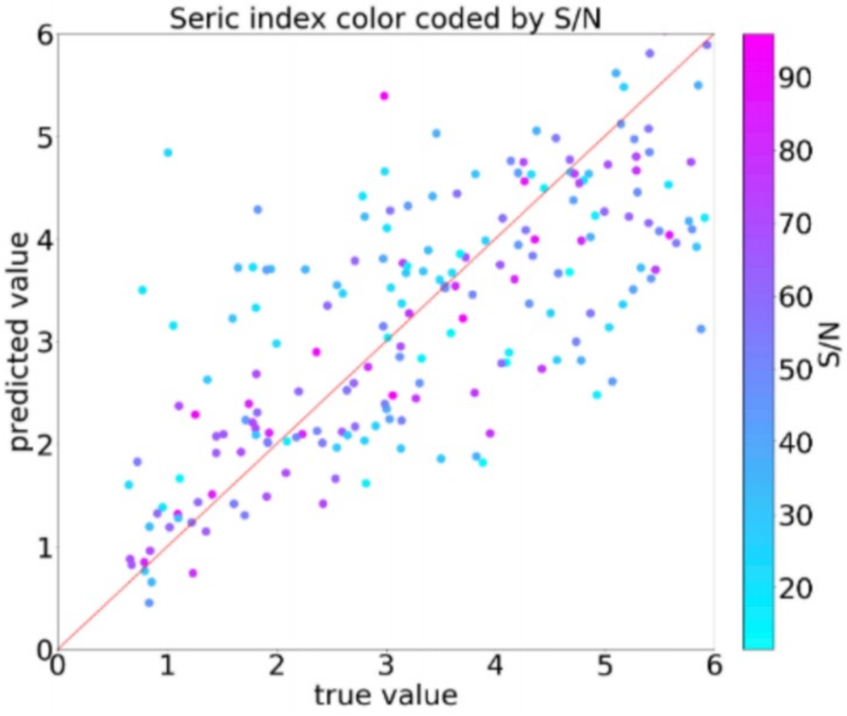
Results of the CNN (5 outputs):

Sersic index — curvature

Ours



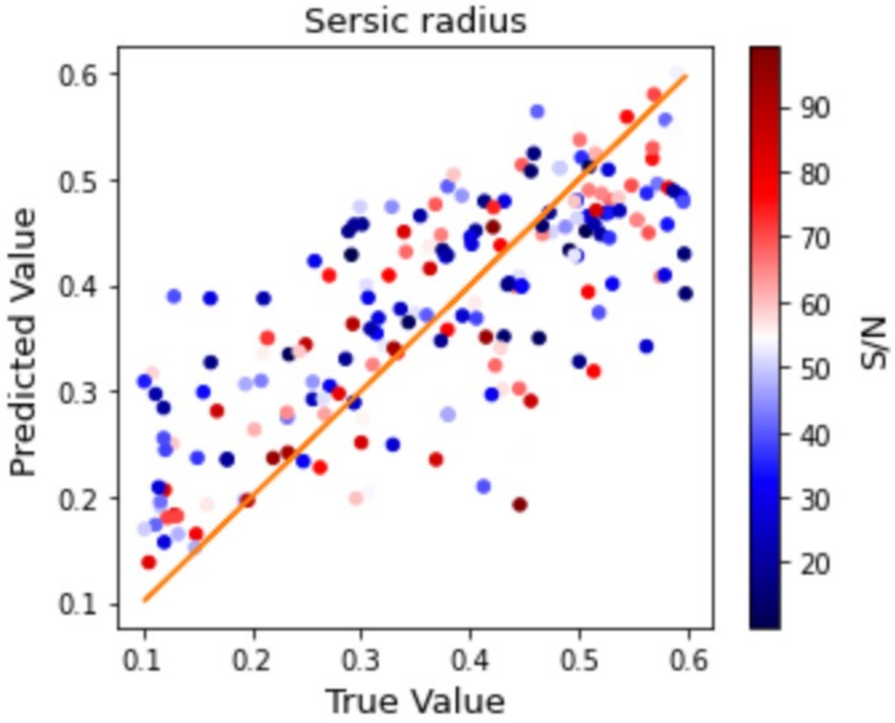
Benchmark



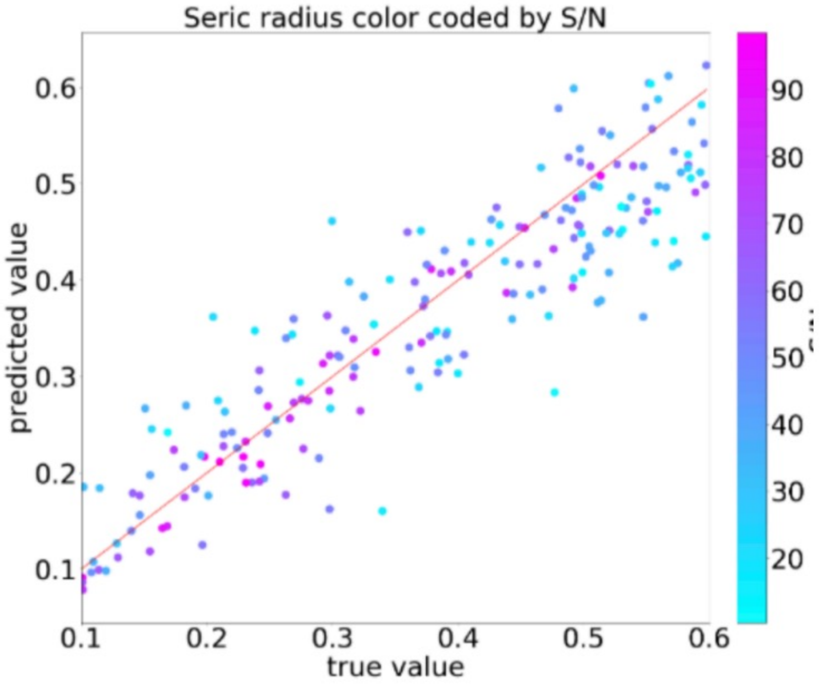
Results of the CNN (5 outputs):

Sersic radius — half-light radius

Ours



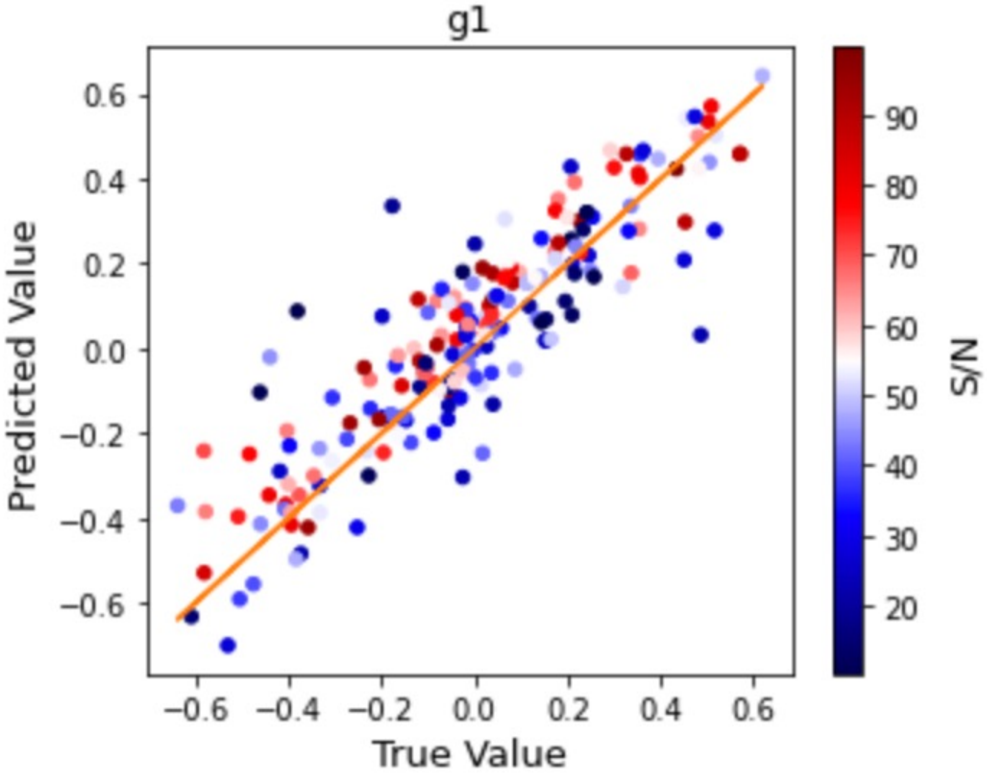
Benchmark



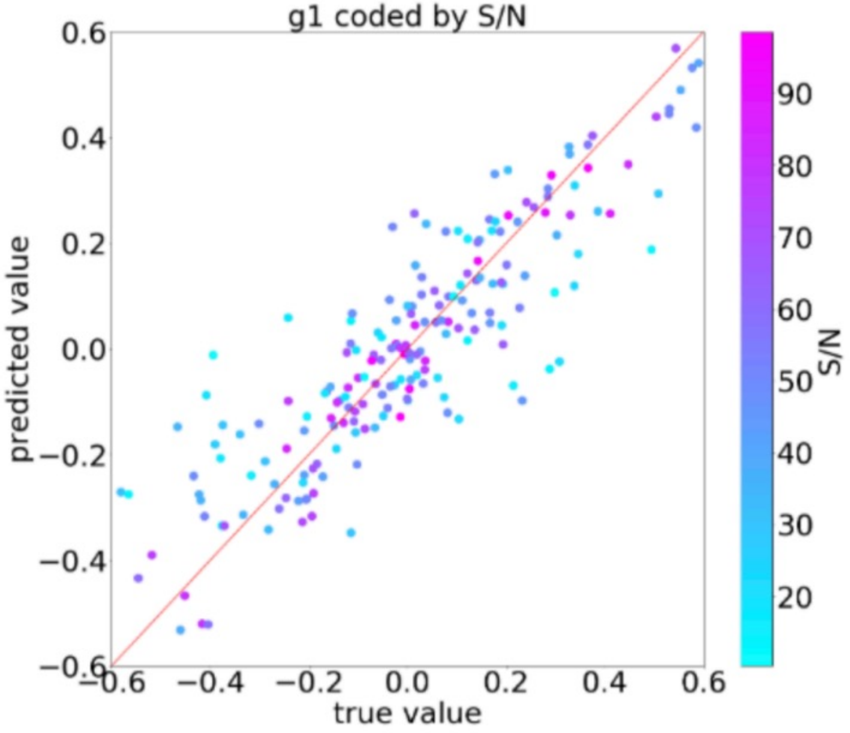
Results of the CNN (5 outputs):

g1 — orientation

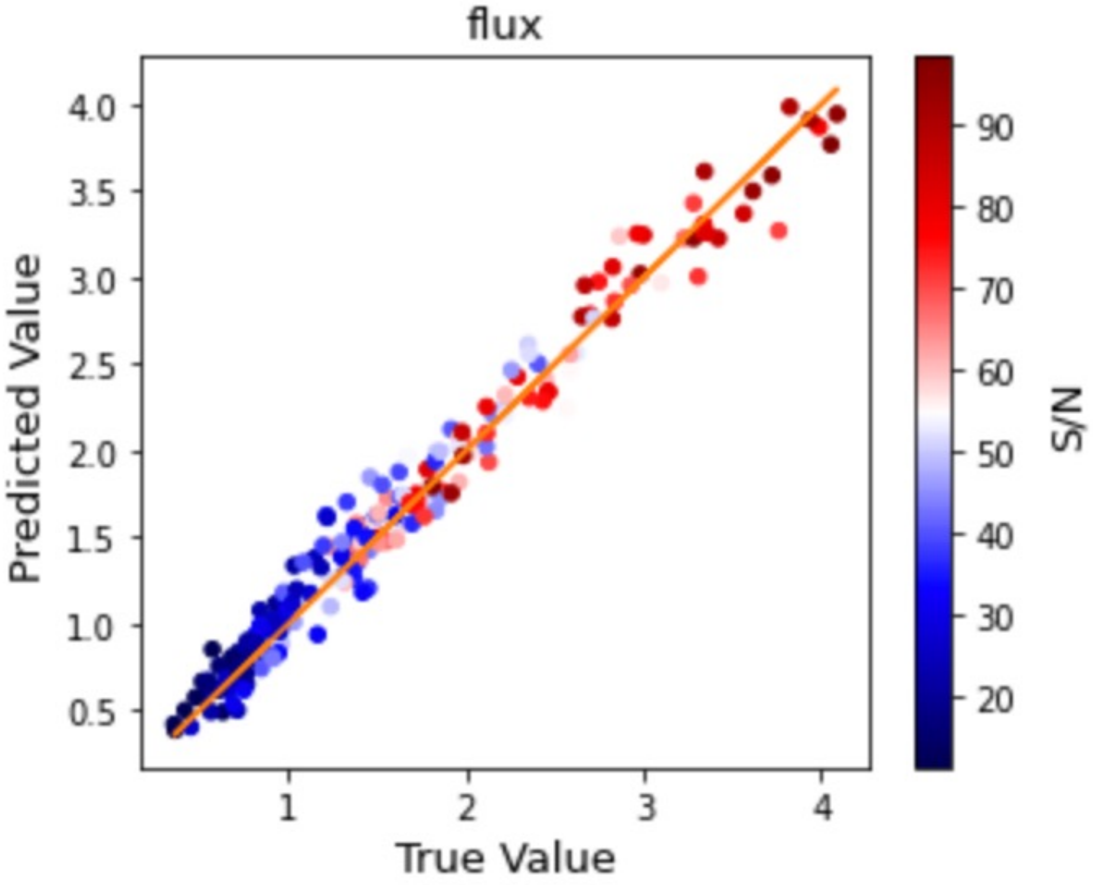
Ours



Benchmark

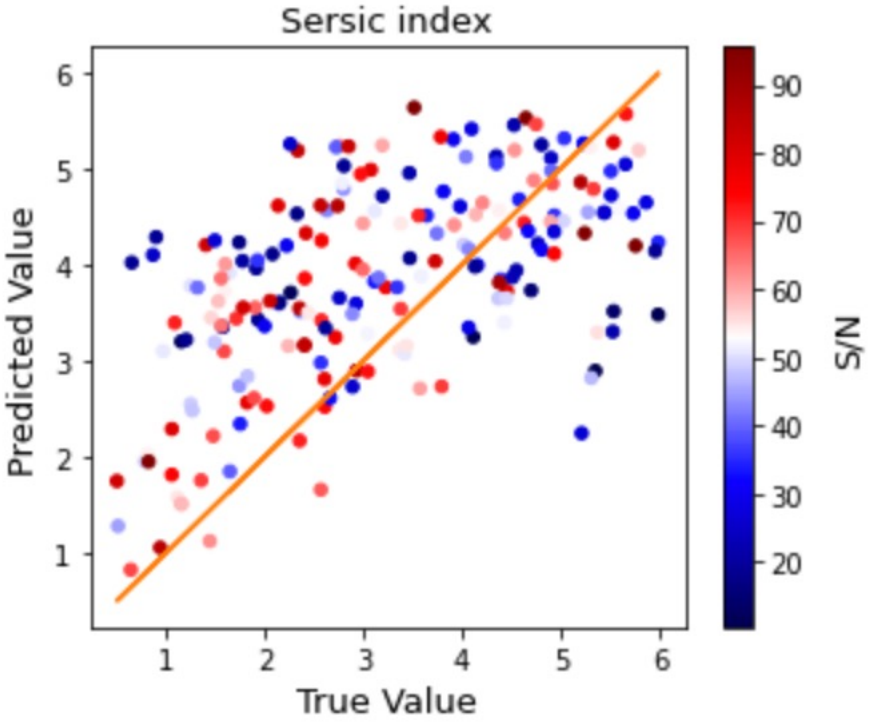


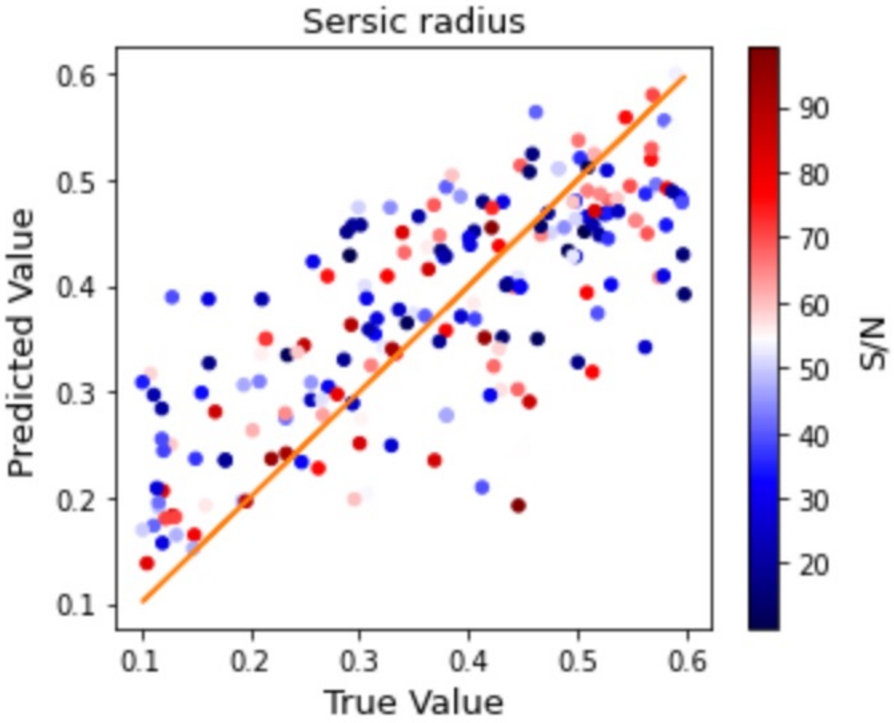
Results of the CNN (5 outputs):



flux —brightness divided by area

Results of the CNN (5 outputs):

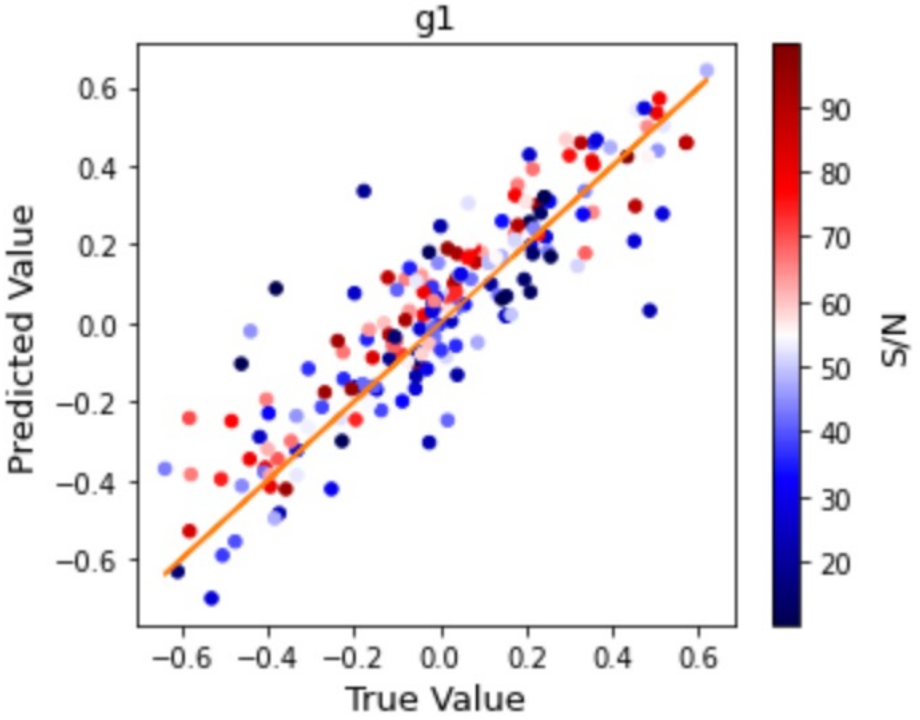




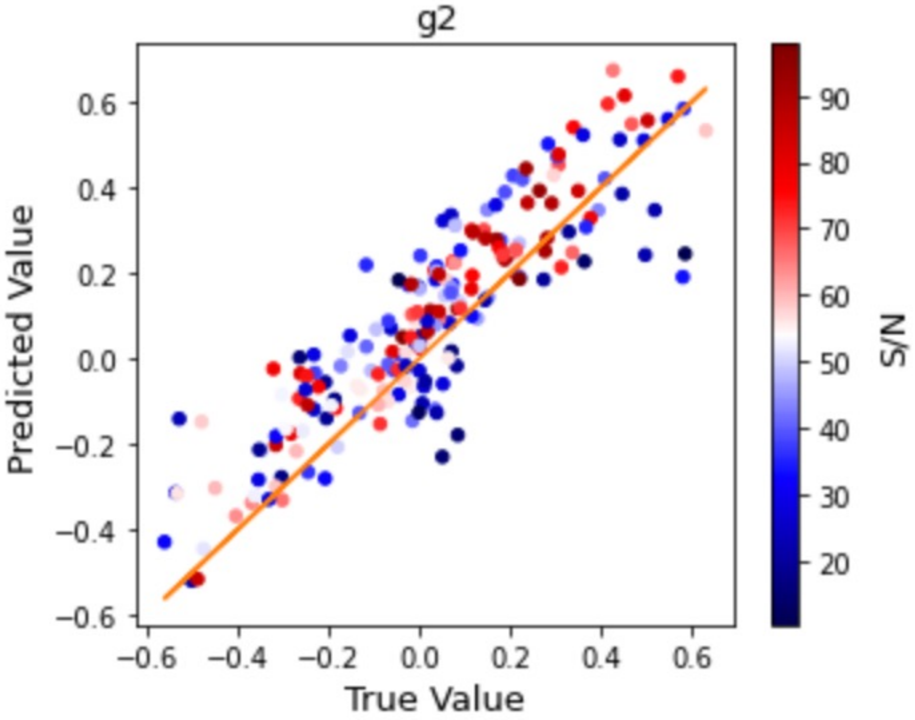
Sersic index — curvature

Sersic radius — half-light radius

Results of the CNN (5 outputs):



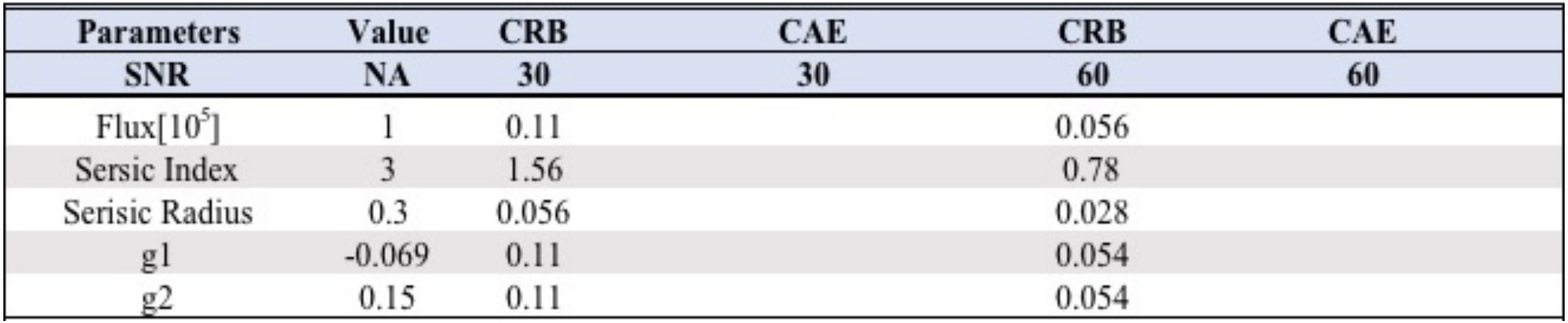
g1 — orientation



g2 —ellipticity

Results of the CNN (5 outputs):

CRAMER-RAO Bound Comparison



Parameters	Value	CRB	CAE	CRB	CAE
SNR	NA	30	30	60	60
Flux[10^5]	1	0.11		0.056	
Sersic Index	3	1.56		0.78	
Serisic Radius	0.3	0.056		0.028	
g1	-0.069	0.11		0.054	
g2	0.15	0.11		0.054	