Galaxy Image Sorting

By Travis Beebe, Evelyn Kimbirk, Michael Hench, Noah Hood, Zice Zhao

Introduction

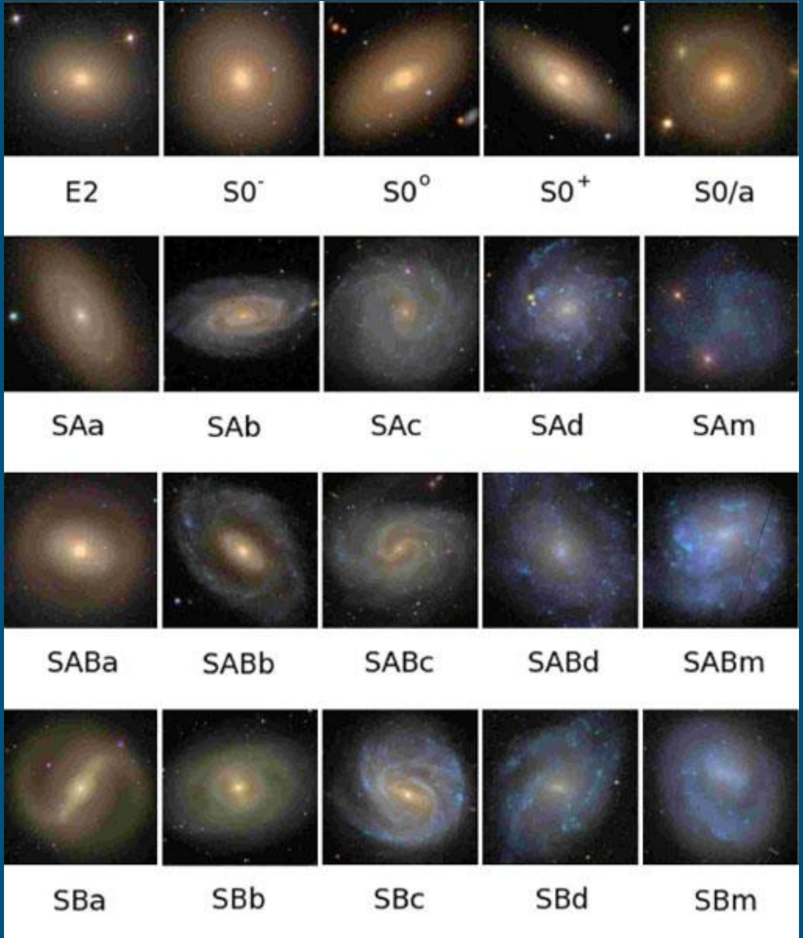
- Space is not as well understood as it should be
- There's too much to process out there
- New technologies will bring in a lot of data about space

Motivation

- Galaxies are popular topics of study

- There are a lot of galaxies to study

- How accurately can a computer sort galaxies?



Background

- Kaggle created a competition about sorting galaxies with ML

- We tried a different approach with a neural network rather than Boosted Decision Trees

Dataset (Noah)

- Kaggle Galaxy Zoo Challenge
- 61,578 training; 79,975 test images
- Each image is 424 by 424 pixels
- 37 galaxy classifiers
    - eg. smooth, has features, is a star, etc.
    - Each class is a prob from 0.0 to 1.0
- Original set classified by volunteers
- Goal is to match classifications

Table 1: The Galaxy Zoo 2 decision tree in words

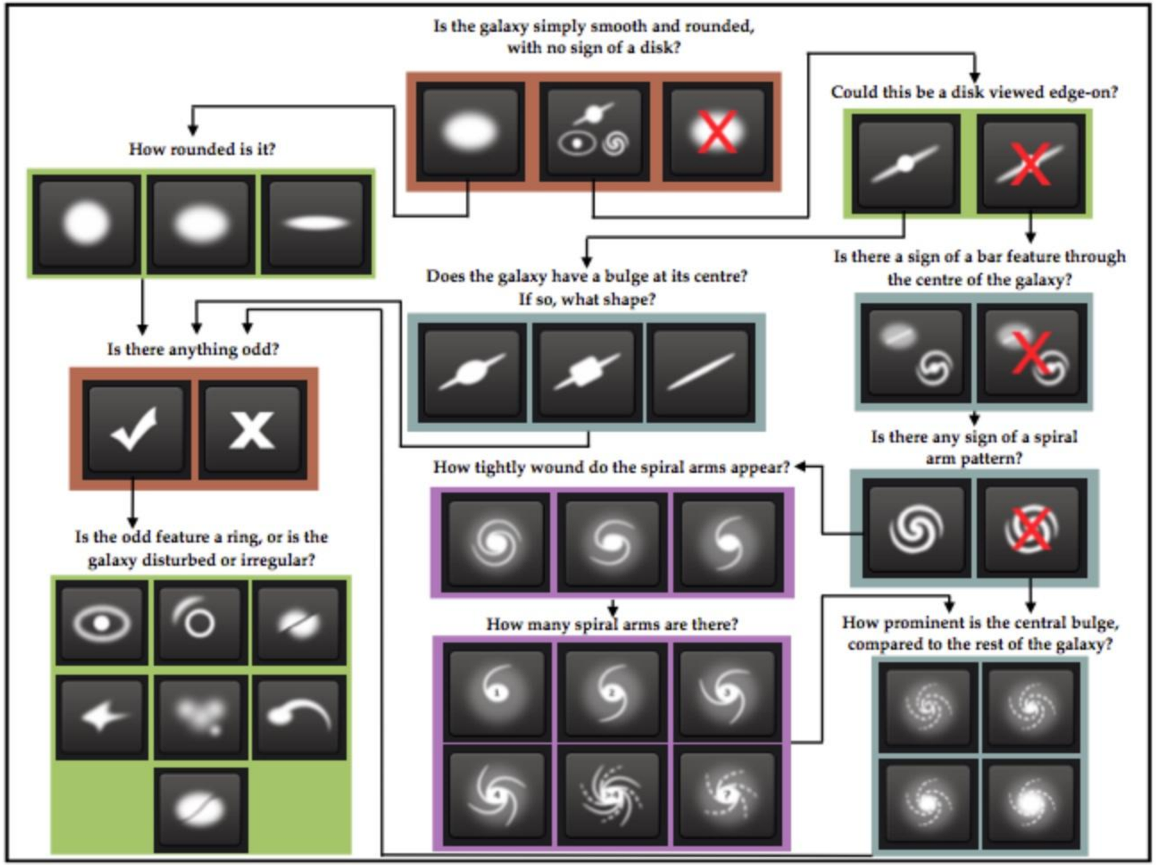
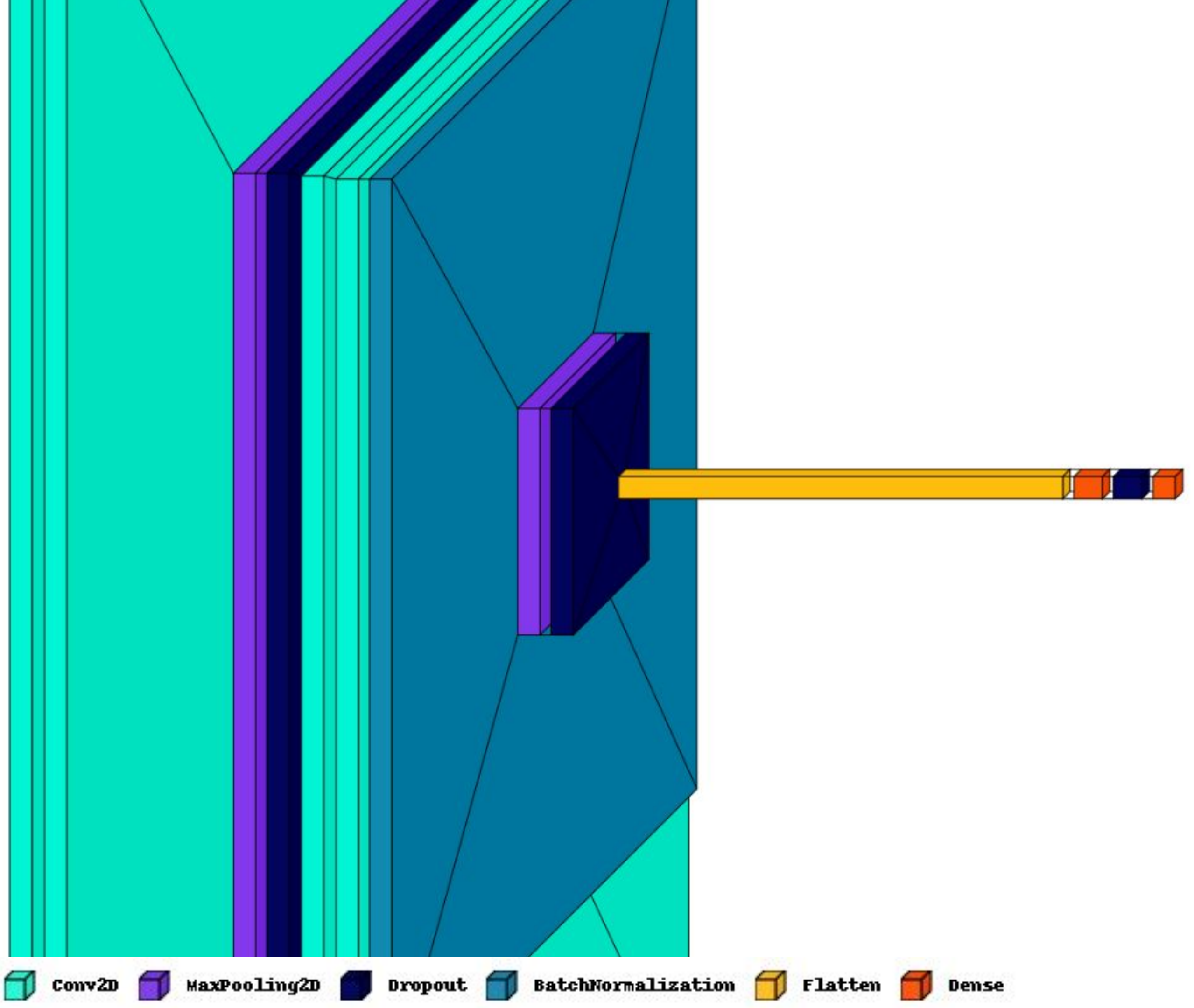

Is the galaxy simply smooth and rounded, with no sign of a disk?
Could this be a disk viewed edge-on?
How rounded is it?
Is there a sign of a bar feature through the centre of the galaxy?
Does the galaxy have a bulge at its centre? If so, what shape?
Is there anything odd?
Is there any sign of a spiral arm pattern?
How tightly wound do the spiral arms appear?
Is the odd feature a ring, or is the galaxy disturbed or irregular?
How many spiral arms are there?
How prominent is the central bulge, compared to the rest of the galaxy?

Figure 1. Flowchart of the classification tasks for GZ2, beginning at the top centre. Tasks are colour-coded by their relative depths in the decision tree. Tasks outlined in brown are asked of every galaxy. Tasks outlined in green, blue, and purple are (respectively) one, two or three steps below branching points in the decision tree. Table 2 describes the responses that correspond to the icons in this diagram.

Methods (Zice)

- VGG6 (Visual Geometry Group)
- Binary Cross Entropy
- Adam



The Model

- 2 convolutional layers of 16 filters and a kernel size of 3x3
- Max pooling layer with a pool size of 2x2
- Dropout layer with a rate of 0.25
- 2 convolutional layers of 32 filters and a kernel size of 3x3
- Max pooling layer with a pool size of 4x4
- Dropout layer with a rate of 0.25
- Flatten layer which flattens the input into a one-dimensional vector
- Dense layer with 256 units
- Dropout layer with a rate of 0.5
- Output dense layer with 37 units, one for each class, with sigmoid activation

Evaluation Criteria

- Root Mean Squared Error
- Compare against submissions to the Galaxy Zoo - The Galaxy Challenge

Results (Evelyn)

We will evaluate our model using 3 metrics:

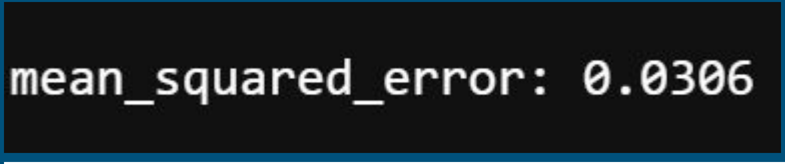
- Root Mean Square Error
- Accuracy
- Area under ROC Curve

The first two metrics will be calculated using the outputs of our model when training was complete.

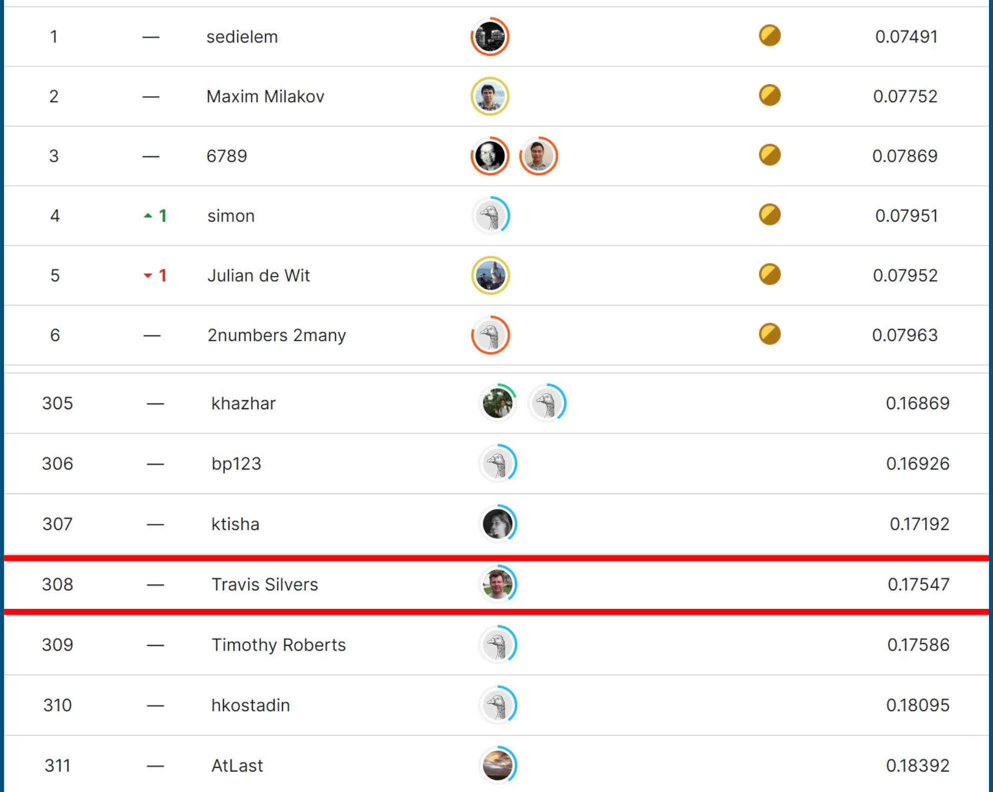
The third metric will use test data that the model has not seen before

RMSE

- On the Leaderboards for the Galaxy Challenge, the top performing models had a RMSE of around 0.075 (when performed on 75% of test data)
- Our model had an RMSE of about 0.17493 (on our training data)
- This would put us at about 308th place on the leaderboards



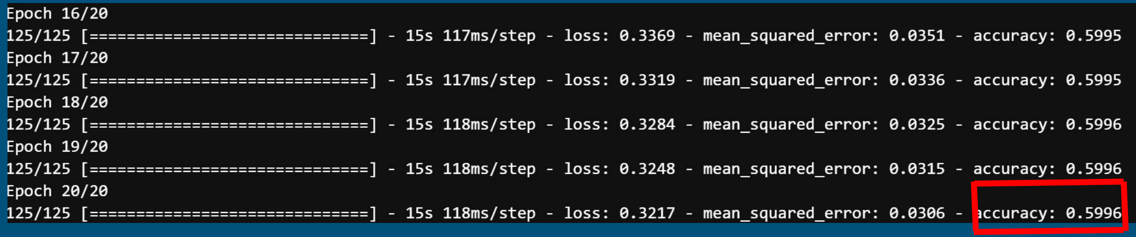
```
mean_squared_error: 0.0306
```



1	—	sedielem	●	0.07491
2	—	Maxim Milakov	●	0.07752
3	—	6789	●	0.07869
4	▲ 1	simon	●	0.07951
5	▼ 1	Julian de Wit	●	0.07952
6	—	2numbers 2many	●	0.07963
305	—	khazhar		0.16869
306	—	bp123		0.16926
307	—	ktisha		0.17192
308	—	Travis Silvers		0.17547
309	—	Timothy Roberts		0.17586
310	—	hkostadin		0.18095
311	—	AtLast		0.18392

Accuracy

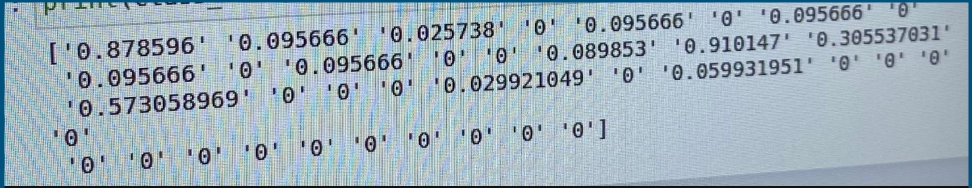
- After training, the model produced an accuracy of about 60% on our training data
- Slightly better than guessing, but not that great at predicting the labels



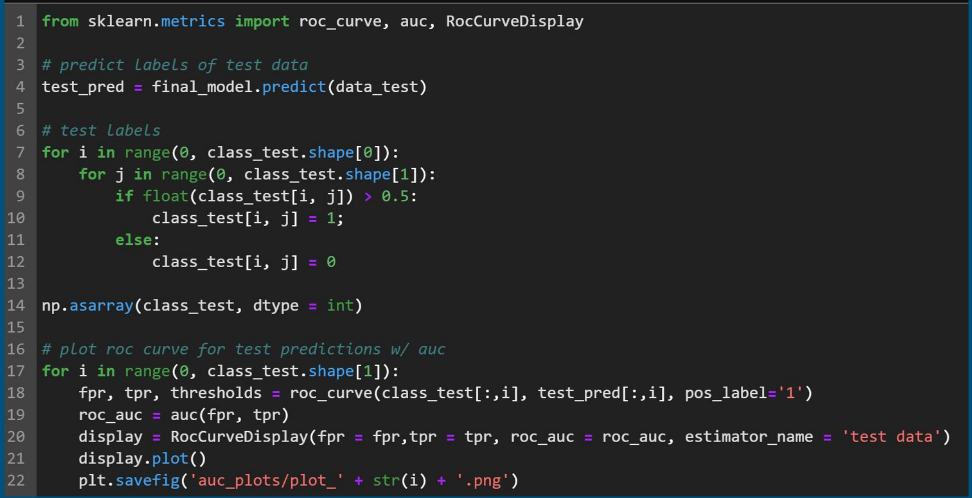
```
Epoch 16/20
125/125 [==============================] - 15s 117ms/step - loss: 0.3369 - mean_squared_error: 0.0351 - accuracy: 0.5995
Epoch 17/20
125/125 [==============================] - 15s 117ms/step - loss: 0.3319 - mean_squared_error: 0.0336 - accuracy: 0.5995
Epoch 18/20
125/125 [==============================] - 15s 118ms/step - loss: 0.3284 - mean_squared_error: 0.0325 - accuracy: 0.5996
Epoch 19/20
125/125 [==============================] - 15s 118ms/step - loss: 0.3248 - mean_squared_error: 0.0315 - accuracy: 0.5996
Epoch 20/20
125/125 [==============================] - 15s 118ms/step - loss: 0.3217 - mean_squared_error: 0.0306 - accuracy: 0.5996
```

ROC Curve

- First, our model was used to predict the labels for test data
- Then each label in the test data was reassigned to be 1 if > 0.5 and 0 if < 0.5
- An ROC Curve was generated for all 37 labels, with the AUC displayed
- Therefore, each ROC Curve is determining how well our model is able to determine whether the majority of people identified a galaxy of belonging to a certain category



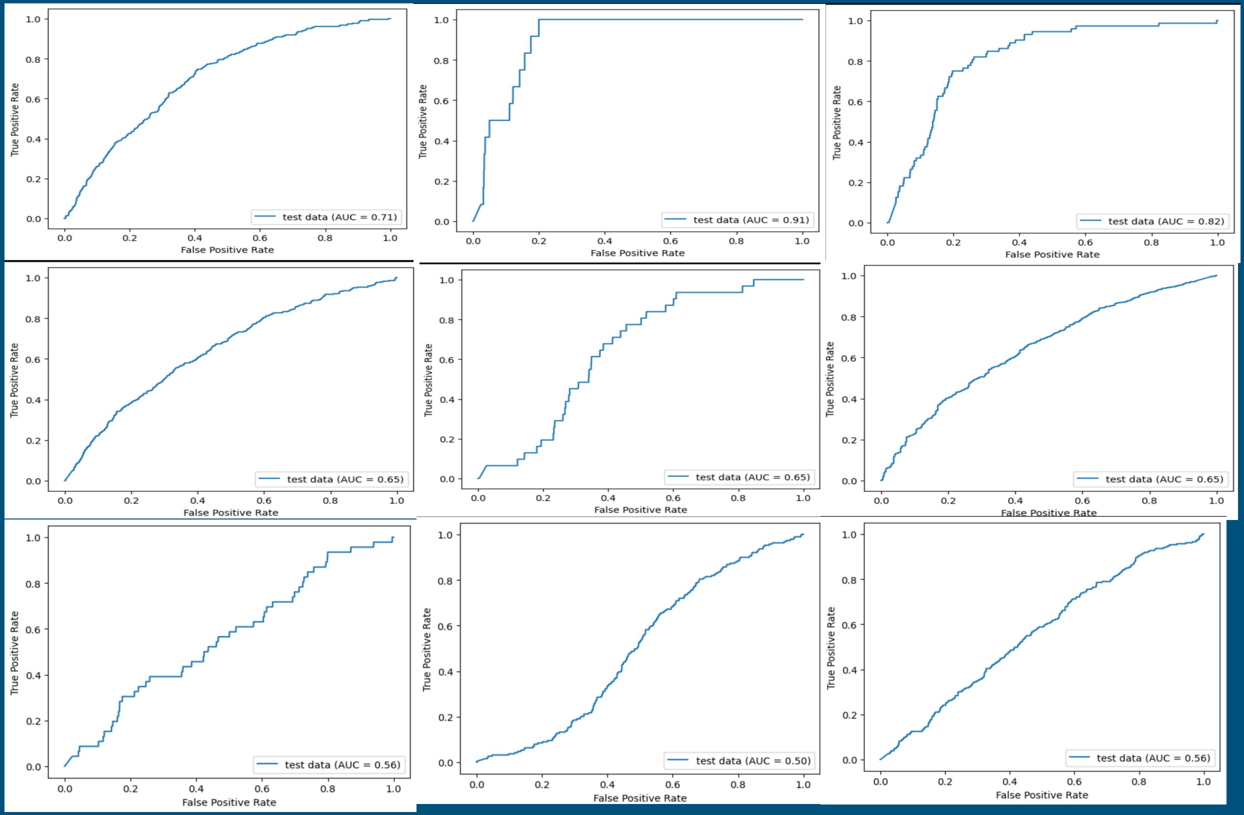
```
['0.878596' '0.095666' '0.025738' '0' '0.095666' '0' '0.095666' '0'
 '0.095666' '0' '0.095666' '0' '0' '0.089853' '0.910147' '0.305537031'
 '0.573058969' '0' '0' '0' '0.029921049' '0' '0.059931951' '0' '0' '0'
'0'
 '0' '0' '0' '0' '0' '0' '0' '0' '0' '0']
```



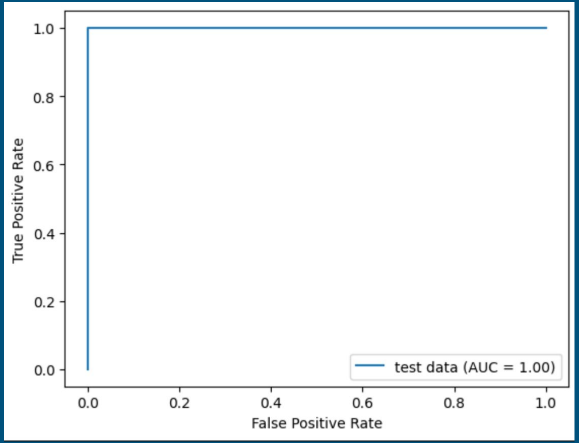
```python
from sklearn.metrics import roc_curve, auc, RocCurveDisplay

# predict labels of test data
test_pred = final_model.predict(data_test)

# test labels
for i in range(0, class_test.shape[0]):
    for j in range(0, class_test.shape[1]):
        if float(class_test[i, j]) > 0.5:
            class_test[i, j] = 1;
        else:
            class_test[i, j] = 0

np.asarray(class_test, dtype = int)

# plot roc curve for test predictions w/ auc
for i in range(0, class_test.shape[1]):
    fpr, tpr, thresholds = roc_curve(class_test[:,i], test_pred[:,i], pos_label='1')
    roc_auc = auc(fpr, tpr)
    display = RocCurveDisplay(fpr = fpr,tpr = tpr, roc_auc = roc_auc, estimator_name = 'test data')
    display.plot()
    plt.savefig('auc_plots/plot_' + str(i) + '.png')
```

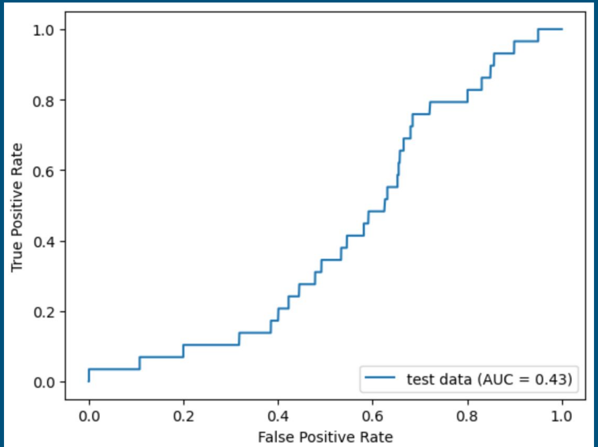
ROC Cont.

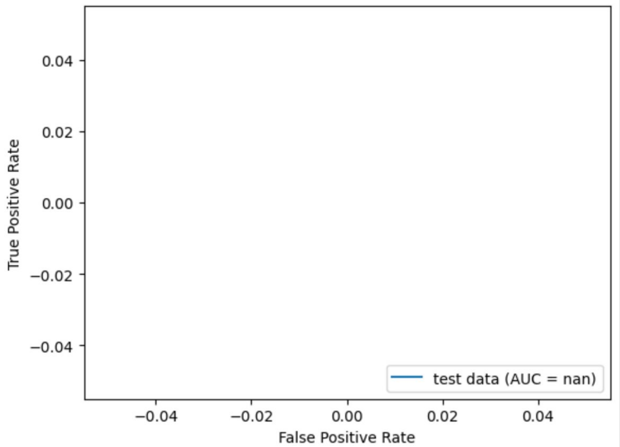
- 19 of the categories had an AUC of over 70%
- 7 had an AUC in the 60-69% range
- 3 had an AUC in the 50-59% range (not much better than randomly guessing the labels)
- Some produced weird results (AUC of 1, NaN, AUC < 50%)

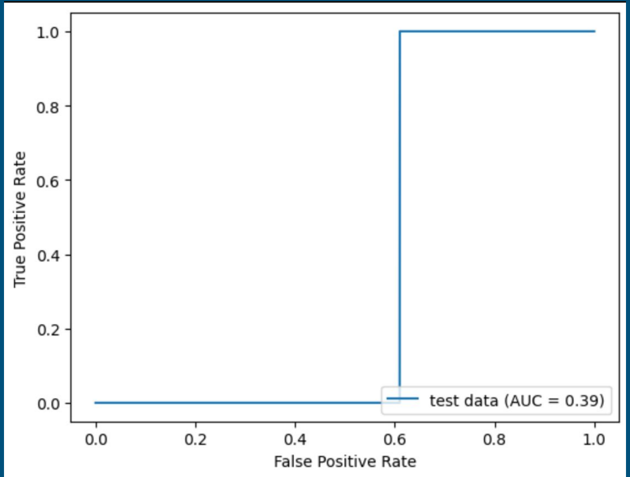


ROC Cont


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
test data (AUC = 1.00)


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
test data (AUC = 0.43)


0.04
0.02
0.00
−0.02
−0.04
True Positive Rate
−0.04
−0.02
0.00
0.02
0.04
False Positive Rate
test data (AUC = nan)


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
test data (AUC = 0.39)

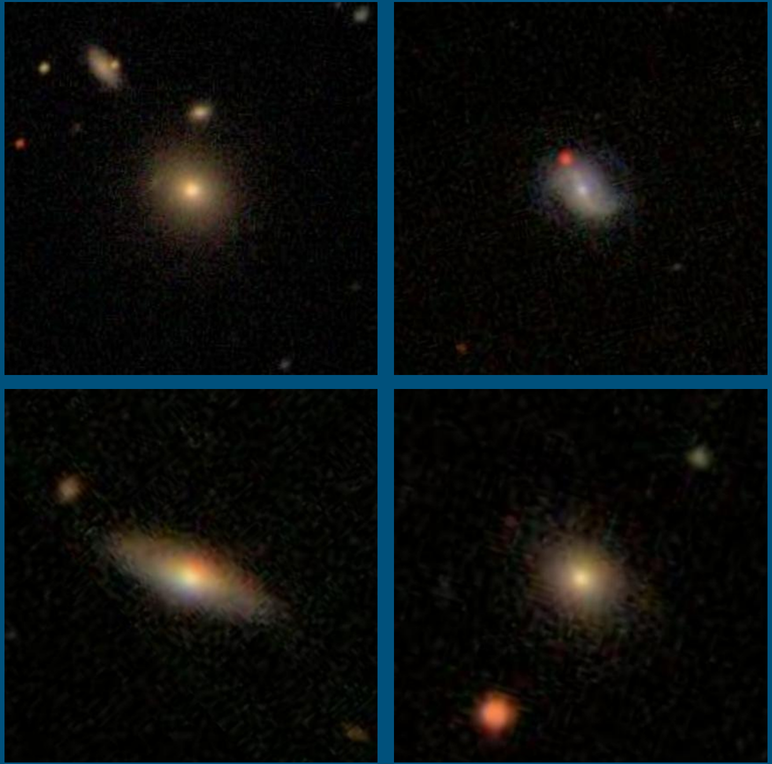
To Simplify…

- The Model is slightly better than randomly guessing, and even does really well in predicting particular categories that a galaxy image would fall into
- However, it is not as accurate as one would hope, and there are significantly better models that yield better results

Conclusion

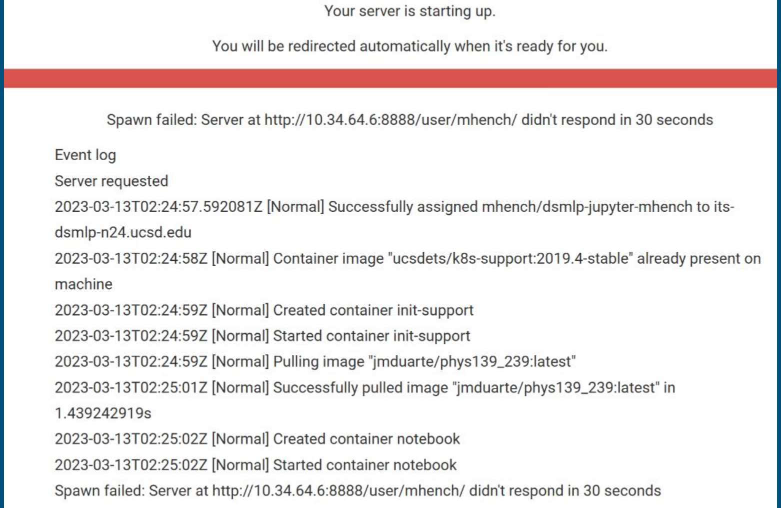
Summary:

- Tasked with classification of 79,975 424x424 images of galaxies
    - 61,578 training images, 37 classes
- Used VGG6 algorithm for its good balance of performance and computational intensity
- Model had ~60% accuracy rating
    - Not great, but not terrible
- Several technical limitations limited our ability to improve the accuracy of our neural network



Limitations

- DataHub
  - Very unstable: kernel constantly dying, server timeouts, sometimes wouldn't even load at all
  - Greatly impacted our ability to effectively create and test code
- Hardware
  - Very RAM + GPU intensive: was only able to train model with ~10,000 images (~16% of training dataset) before GPU (RTX 3080 Ti) ran out of VRAM
    - Limited to simpler algorithms such as VGG6 (larger algorithms would not even train)



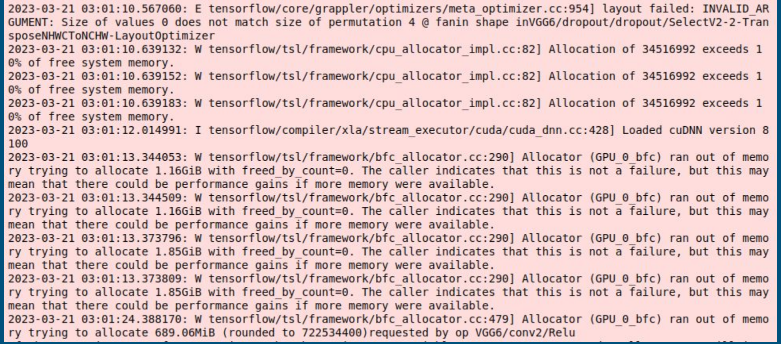
Your server is starting up.

You will be redirected automatically when it's ready for you.

Spawn failed: Server at http://10.34.64.6:8888/user/mhench/ didn't respond in 30 seconds

Event log

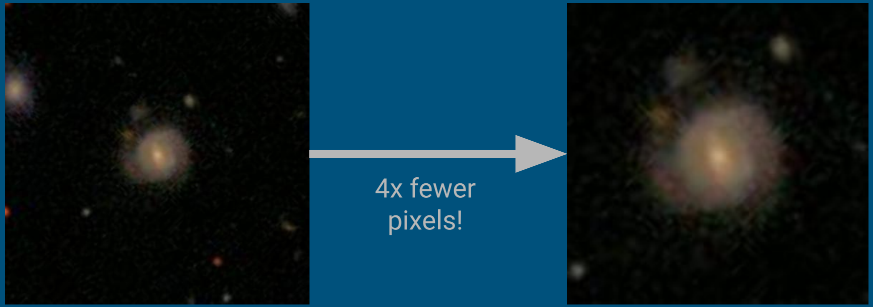
```
Server requested
2023-03-13T02:24:57.592081Z [Normal] Successfully assigned mhench/dsmlp-jupyter-mhench to its-dsmlp-n24.ucsd.edu
2023-03-13T02:24:58Z [Normal] Container image "ucsdets/k8s-support:2019.4-stable" already present on machine
2023-03-13T02:24:59Z [Normal] Created container init-support
2023-03-13T02:24:59Z [Normal] Started container init-support
2023-03-13T02:24:59Z [Normal] Pulling image "jmduarte/phys139_239:latest"
2023-03-13T02:25:01Z [Normal] Successfully pulled image "jmduarte/phys139_239:latest" in 1.439242919s
2023-03-13T02:25:02Z [Normal] Created container notebook
2023-03-13T02:25:02Z [Normal] Started container notebook
Spawn failed: Server at http://10.34.64.6:8888/user/mhench/ didn't respond in 30 seconds
```



```
2023-03-21 03:01:10.567060: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout failed: INVALID_AR
GUMENT: Size of values 0 does not match size of permutation 4 @ fanin shape inVGG6/dropout/dropout/SelectV2-2-Tran
sposeNHWCToNCHW-LayoutOptimizer
2023-03-21 03:01:10.639132: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 34516992 exceeds 1
0% of free system memory.
2023-03-21 03:01:10.639152: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 34516992 exceeds 1
0% of free system memory.
2023-03-21 03:01:10.639183: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 34516992 exceeds 1
0% of free system memory.
2023-03-21 03:01:12.014991: I tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:428] Loaded cuDNN version 8
100
2023-03-21 03:01:13.344053: W tensorflow/tsl/framework/bfc_allocator.cc:290] Allocator (GPU_0_bfc) ran out of memo
ry trying to allocate 1.16GiB with freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-03-21 03:01:13.344509: W tensorflow/tsl/framework/bfc_allocator.cc:290] Allocator (GPU_0_bfc) ran out of memo
ry trying to allocate 1.16GiB with freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-03-21 03:01:13.373796: W tensorflow/tsl/framework/bfc_allocator.cc:290] Allocator (GPU_0_bfc) ran out of memo
ry trying to allocate 1.85GiB with freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-03-21 03:01:13.373809: W tensorflow/tsl/framework/bfc_allocator.cc:290] Allocator (GPU_0_bfc) ran out of memo
ry trying to allocate 1.85GiB with freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-03-21 03:01:24.388170: W tensorflow/tsl/framework/bfc_allocator.cc:479] Allocator (GPU_0_bfc) ran out of memo
ry trying to allocate 689.06MiB (rounded to 722534400)requested by op VGG6/conv2/Relu
```

Potential Improvements/Modifications

- More image preprocessing
  - Image cropping: while images were 424x424, most galaxies were concentrated in the center ~150-200 squared pixels
- Custom model to optimize for accuracy vs computational complexity
  - Don’t have dedicated AI computers with multiple GPUs - have to make as efficient as possible!
- Model compression: only ~15 classes made up vast majority of galaxies!
- Training model on more powerful hardware (esp. more RAM, better GPU)



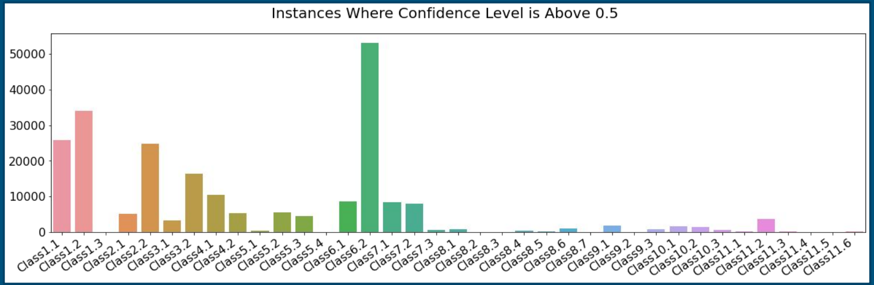


Image credit: https://jayspeidell.github.io/

Thank You!

Travis Beebe - Project proposal, project report

Evelyn Kimbirk - Metrics, debugging, and optimization

Michael Hench - Neural network creation, debugging, and optimization

Noah Hood - Analyzation techniques, data processing

Zice Zhao - Setup, project report

GitHub: https://github.com/blackcomb-dev/phys139-239-final-proj