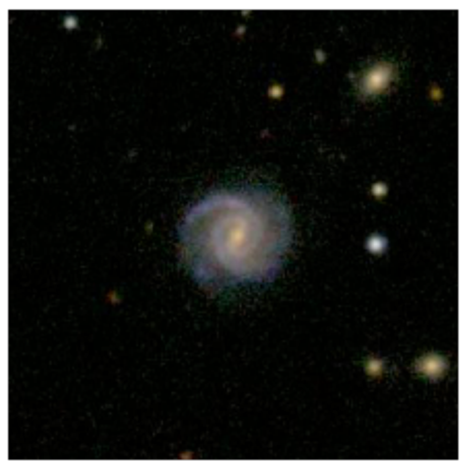
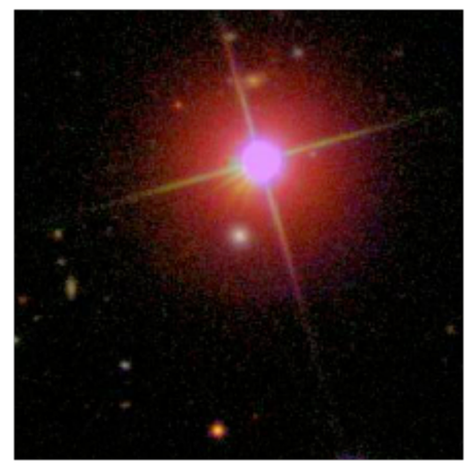
Part II
Automatic classification of galaxies morphology

Júlio Caineta

Goal

- Automatic classification of galaxies morphology based on images labeled by humans

Motivation

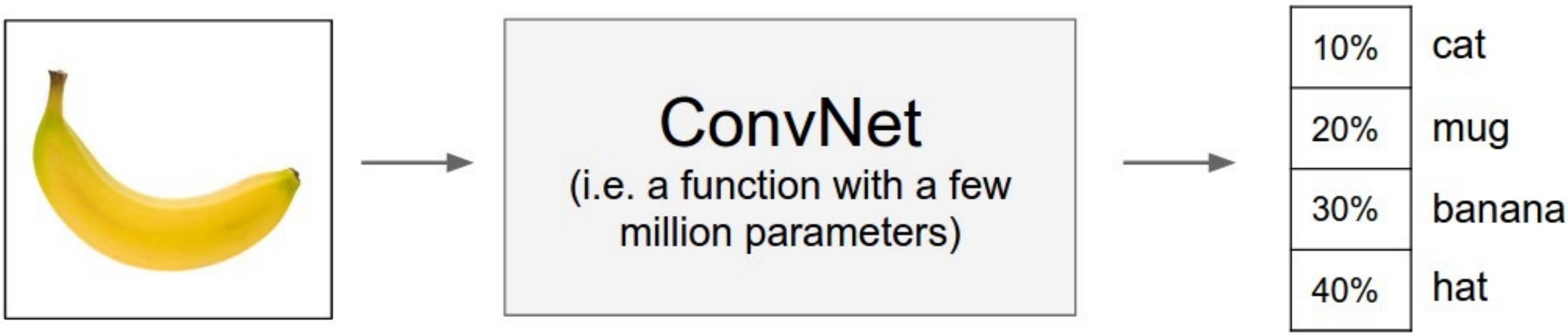
- $2x10^{11}$ to $2x10^{12}$ galaxies
- GZv1: 900k, 50M, 2 years — $4.5x10^{-6}$%

How to perform this task

- Classification / Regression
- Images are not exactly data points
- Computer Vision

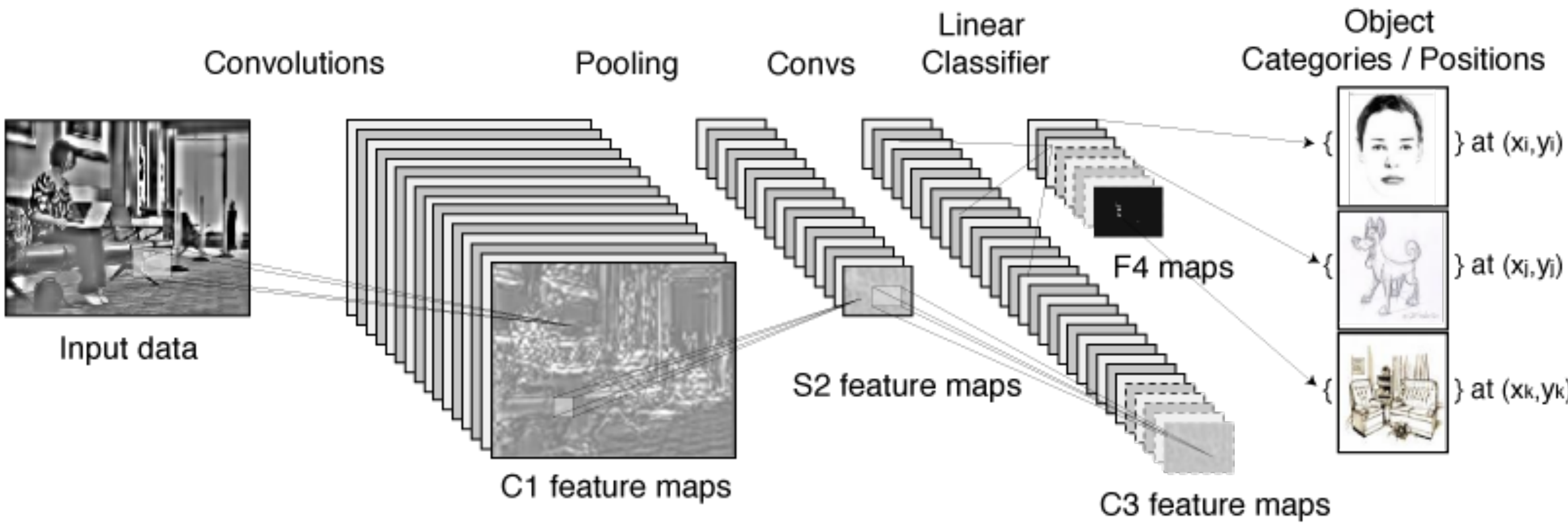
Convolutional Neural Networks

What are ConvNets



Karpathy, 2015 (http://karpathy.github.io)

What are ConvNets

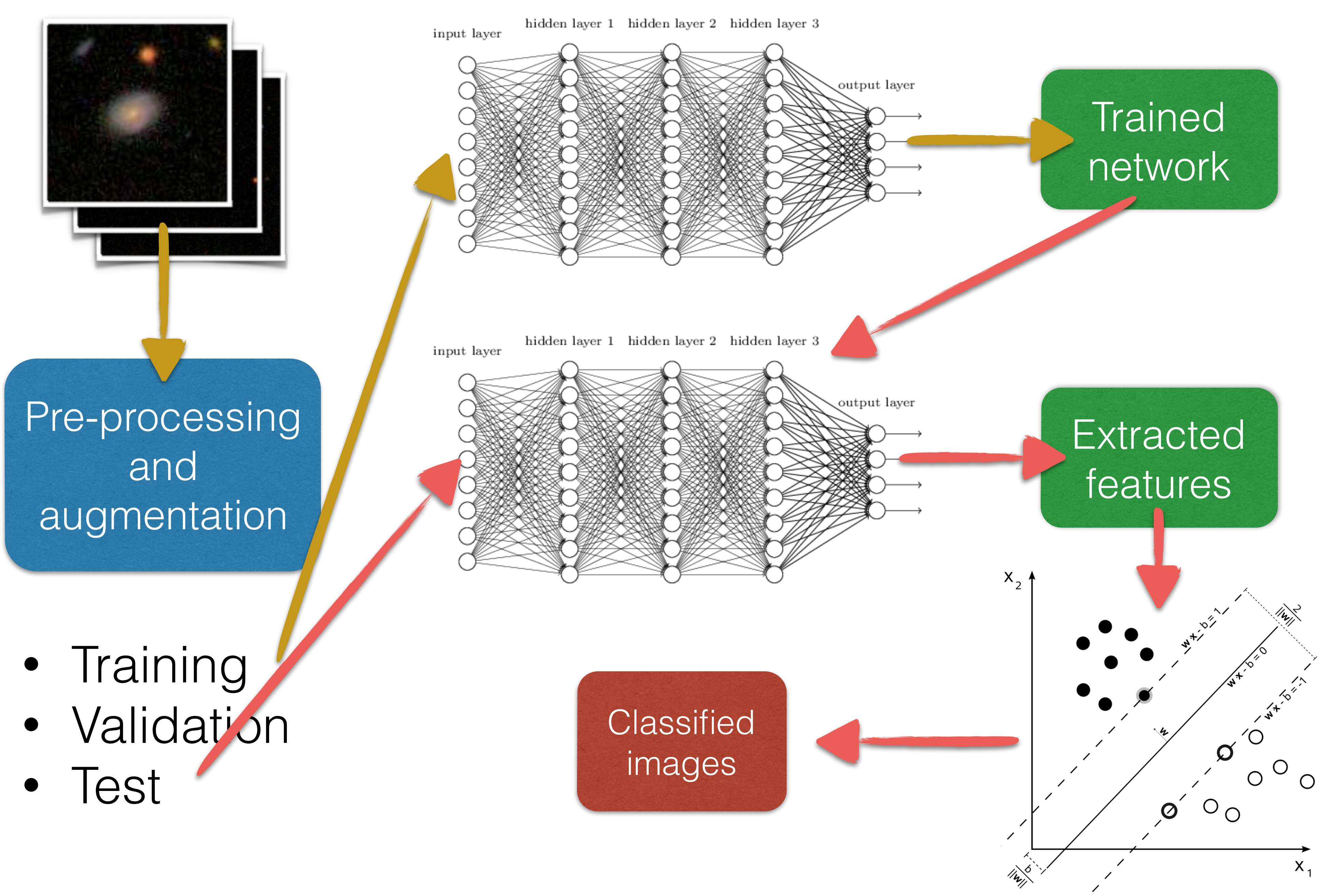


Clément Farabet, 2011

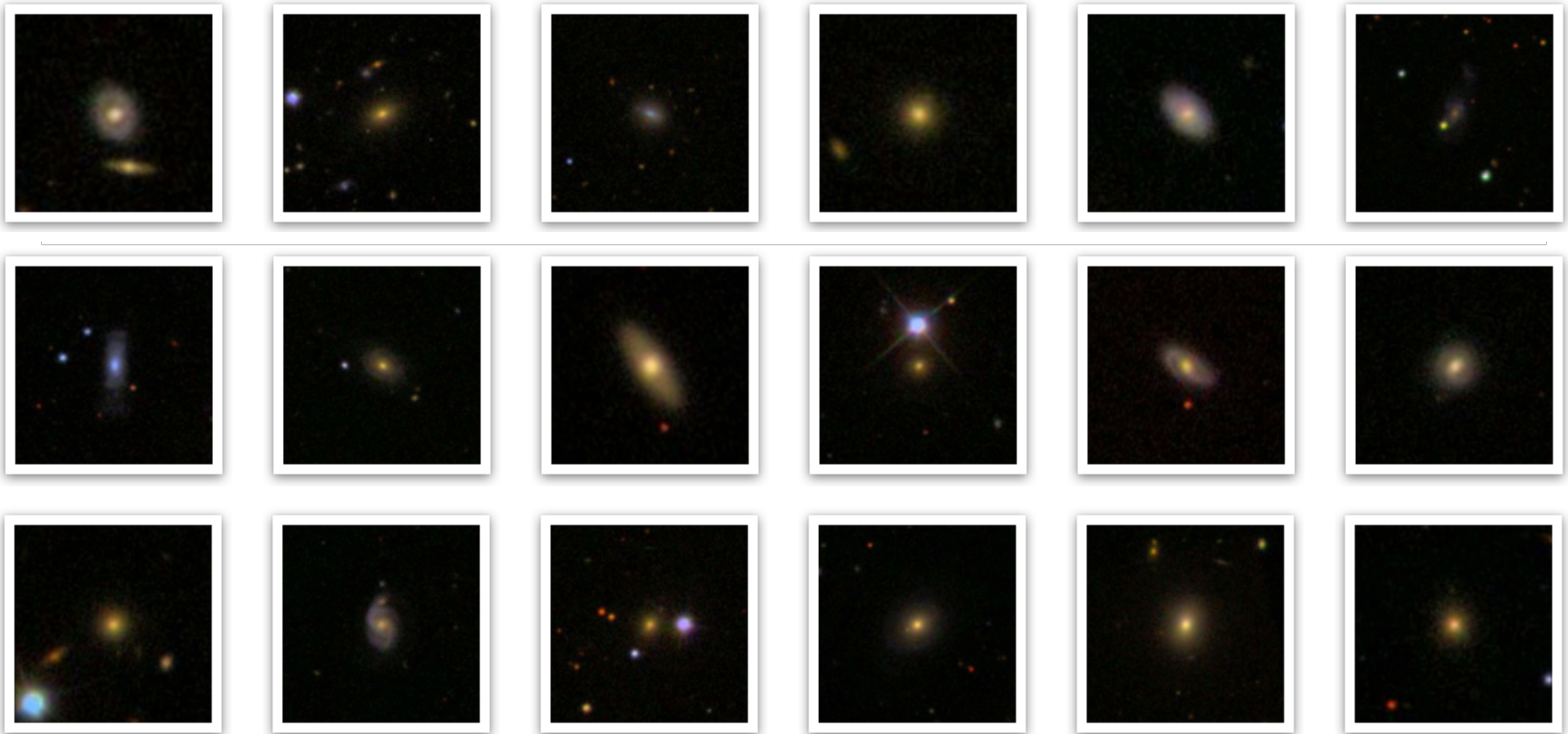
Where do we come from and where are we going

- Yearly conferences / competitions on CV
- Kaggle competition with more than 300 submissions
    - > 200 000 brightest Sloan galaxies
    - 60 millions classifications over 14 months

Workflow


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer
Trained network
input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer
Extracted features
Pre-processing and augmentation
Training
Validation
Test
Classified images
x_2
x_1

Data preparation

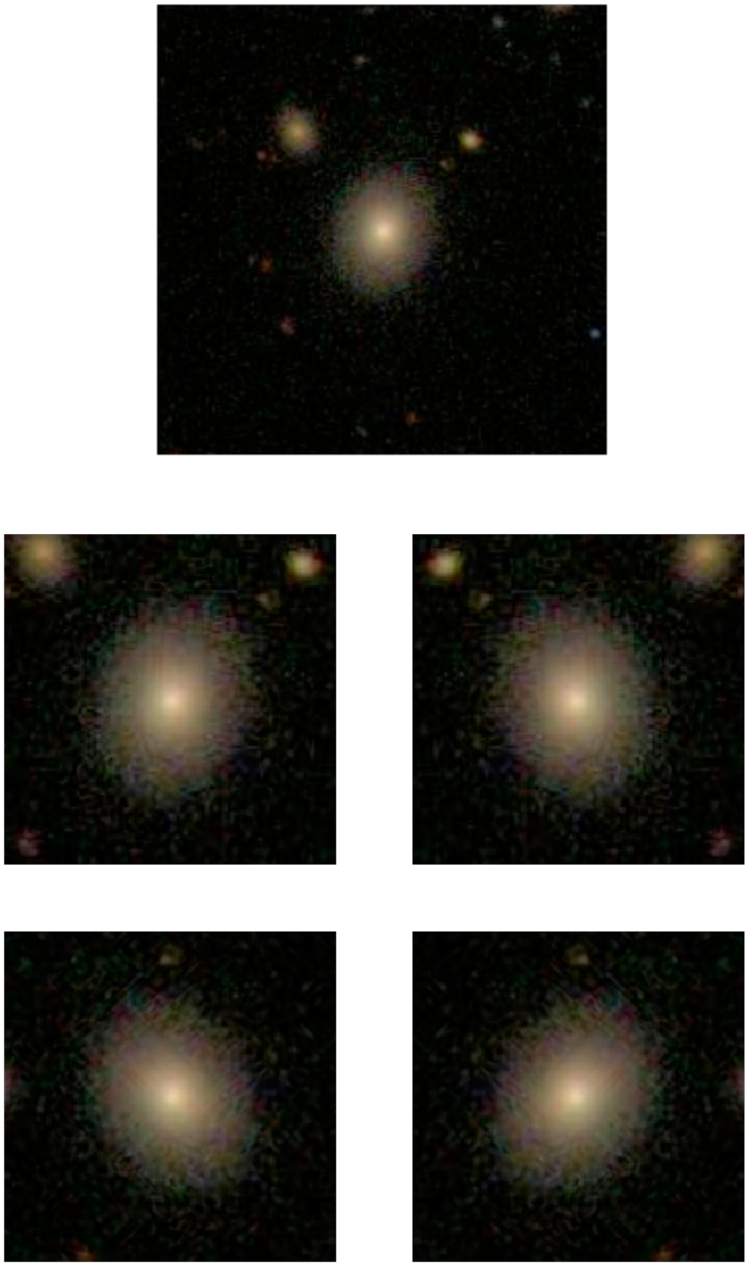


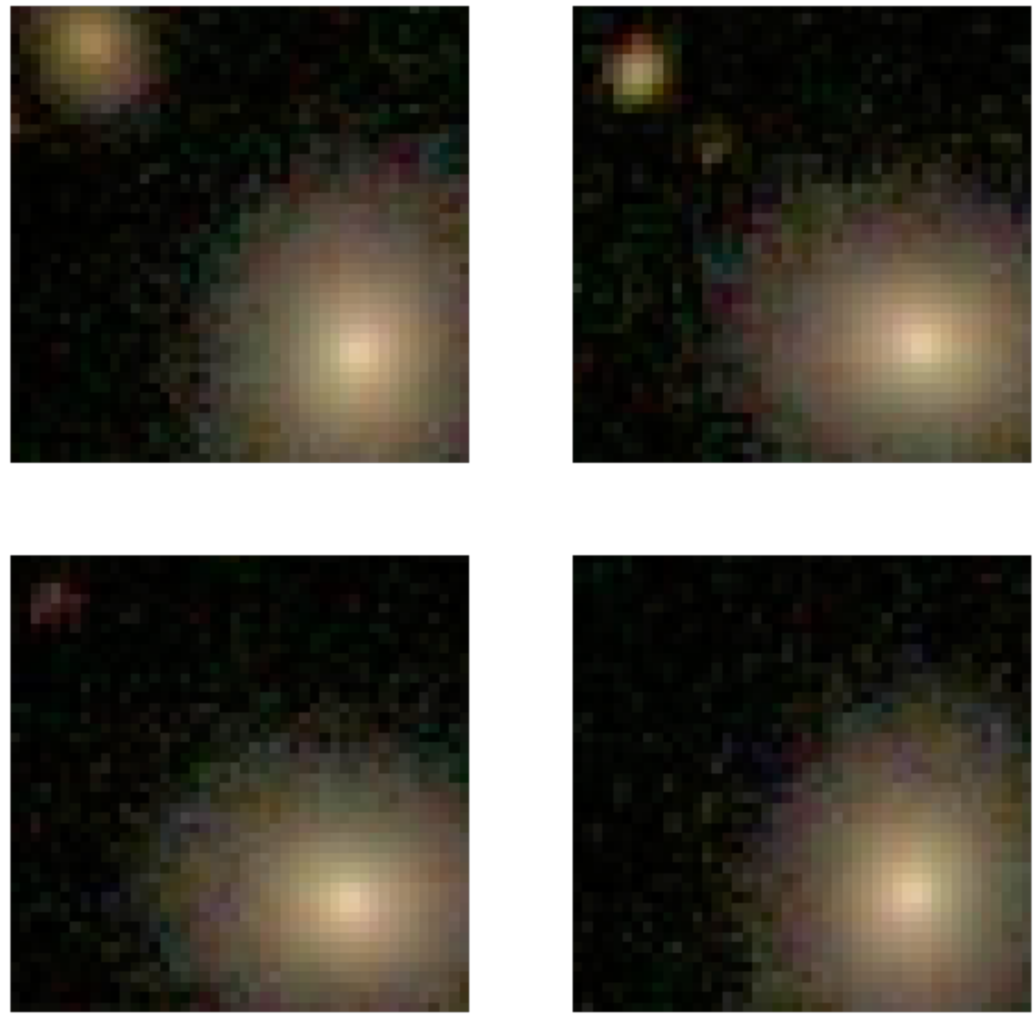
kaggle

61578

GALAXY ZOO

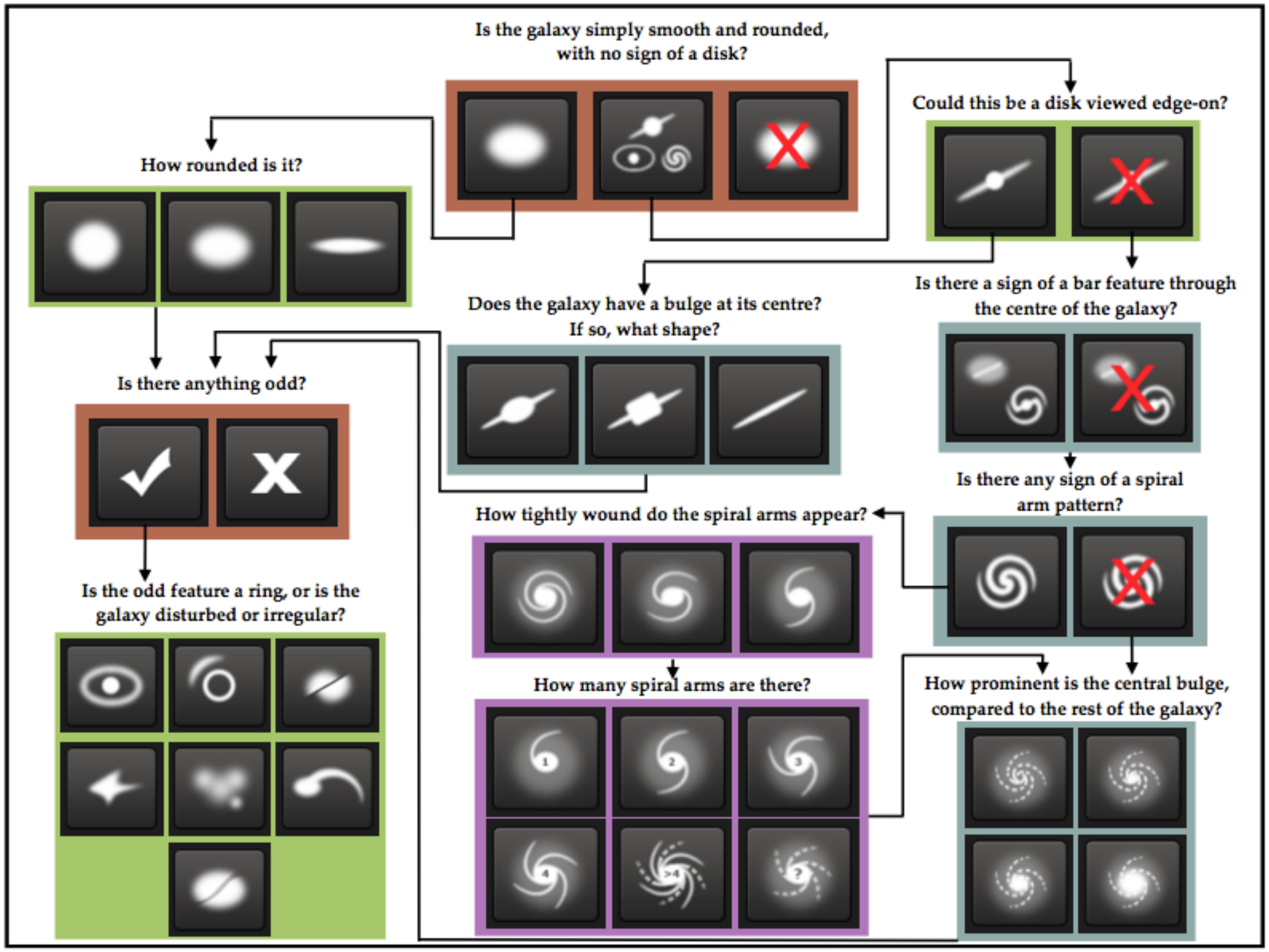
crop + rotate 45 + flip
downsample + color noise
overlapping





x16 = 985 248

Classifications



Classifications

```
In [199]: sol.shape
Out[199]: (61578, 38)
```

```
In [3]: sol.head()
Out[3]:
```

	GalaxyID	Class1.1	Class1.2	Class1.3	Class2.1	Class2.2	Class3.1	Class3.2	Class4.1	Class4.2	...	Class9.3	Class10.1	Class10.2	Class10.3
0	100008	0.383147	0.616853	0.000000	0.000000	0.616853	0.038452	0.578401	0.418398	0.198455	...	0.000000	0.279952	0.138445	0.000000
1	100023	0.327001	0.663777	0.009222	0.031178	0.632599	0.467370	0.165229	0.591328	0.041271	...	0.018764	0.000000	0.131378	0.459950
2	100053	0.765717	0.177352	0.056931	0.000000	0.177352	0.000000	0.177352	0.000000	0.177352	...	0.000000	0.000000	0.000000	0.000000
3	100078	0.693377	0.238564	0.068059	0.000000	0.238564	0.109493	0.129071	0.189098	0.049466	...	0.000000	0.094549	0.000000	0.094549
4	100090	0.933839	0.000000	0.066161	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000

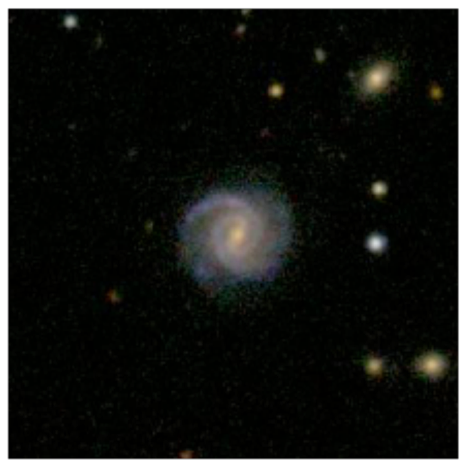
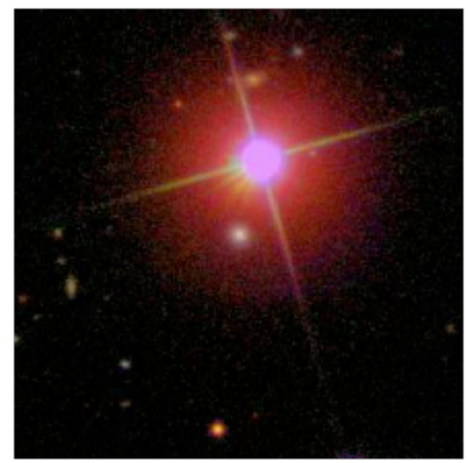
5 rows × 38 columns

```
In [327]: probs = sol.values[3, 1:4]
          probs
Out[327]: array([ 0.693377,  0.238564,  0.068059])
```

```
In [328]: x = np.random.rand(16)
          cond = [x < probs[0], x < sum(probs[:2])]
          classes = [0, 1]
          np.select(cond, classes, 2)
Out[328]: array([2, 0, 0, 0, 0, 1, 2, 0, 0, 2, 0, 0, 0, 1, 1, 0])
```

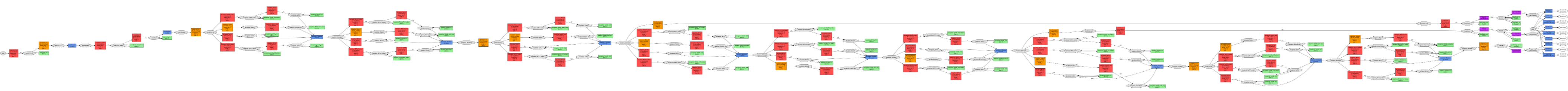
Classifications

Q1. Is the object a smooth galaxy, a galaxy with features/disk or a star?

Smoothed/Rounded	Features/Disk	Artifact/Star
		
100%	100%	93%

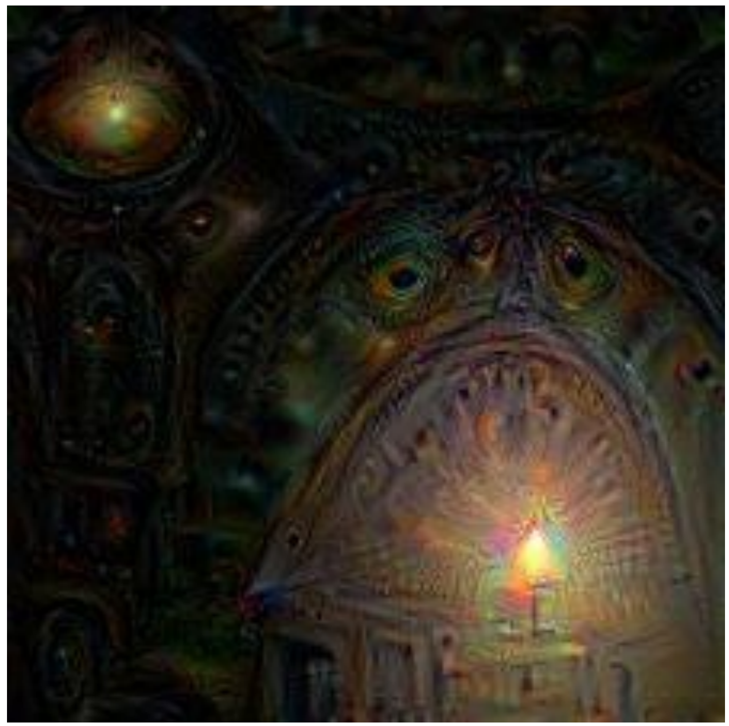
ConvNet

- Architecture: GoogLeNet (2014)



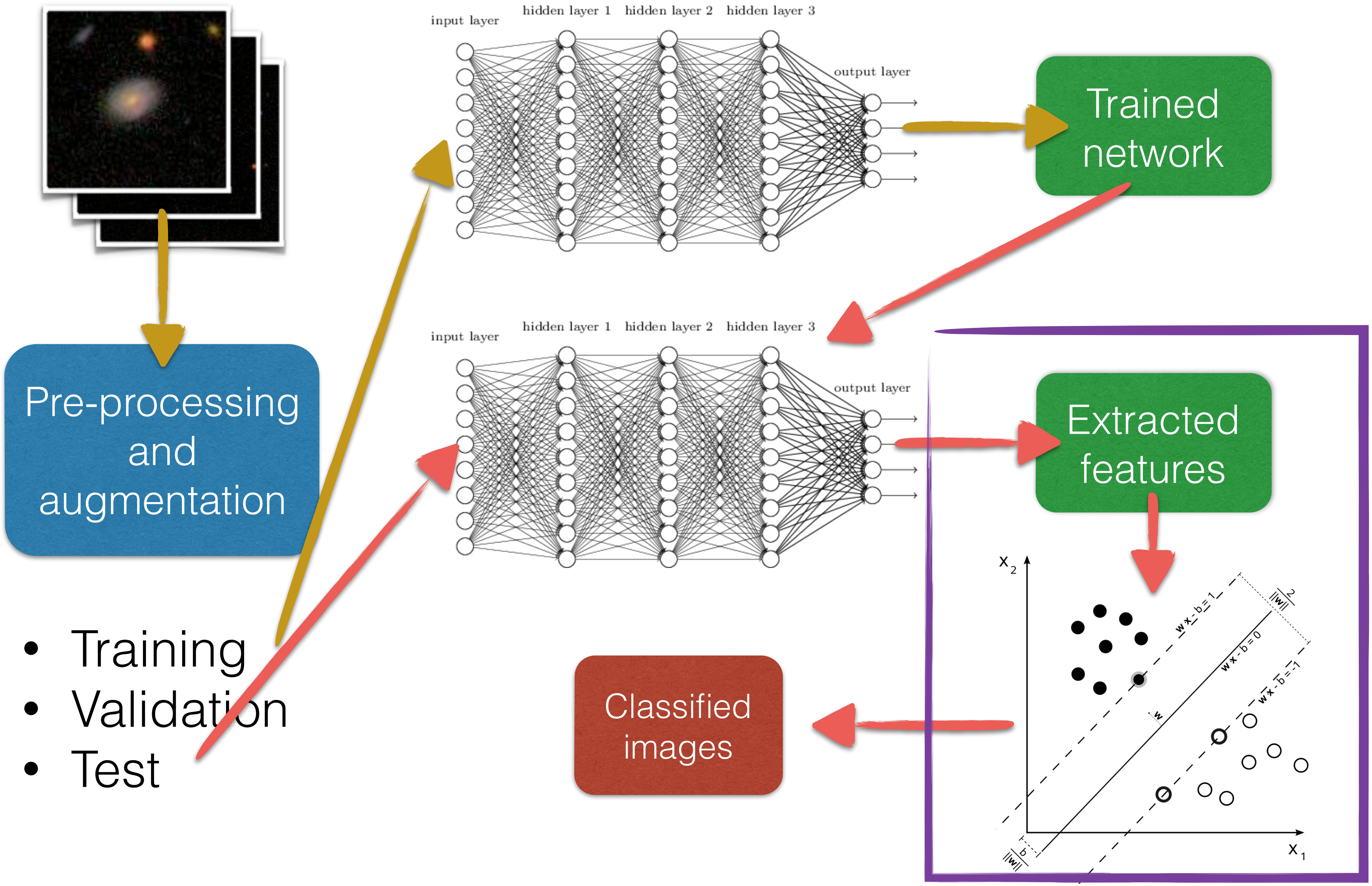
- Train our own network (on the pre-trained network)
- 30 epochs
- Run again with the test set
- Feature extractor

Extracted features



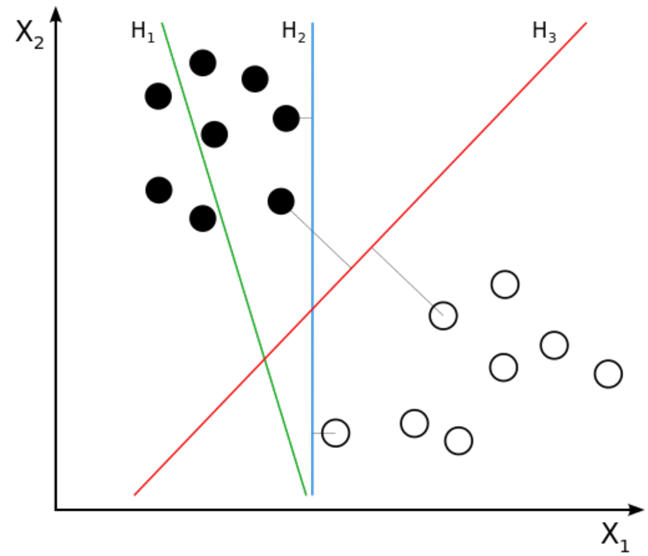
Google's Deepdream (https://github.com/google/deepdream)
Bat-country (https://github.com/jrosebr1/bat-country)

Workflow



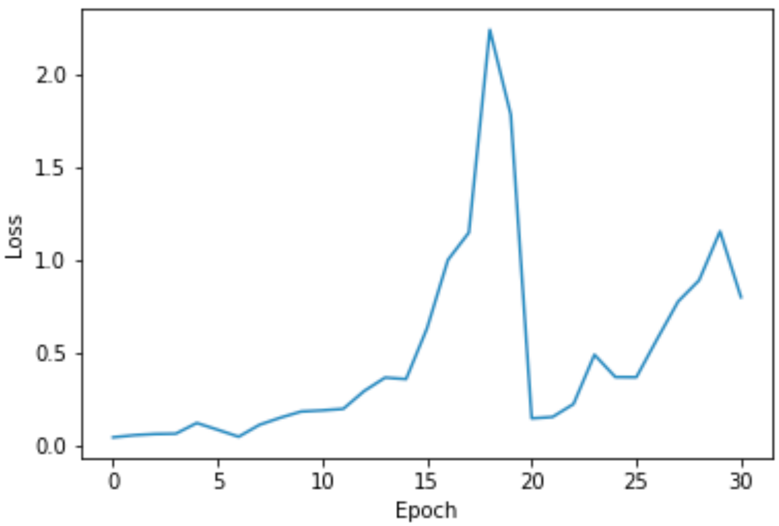
- Training
- Validation
- Test

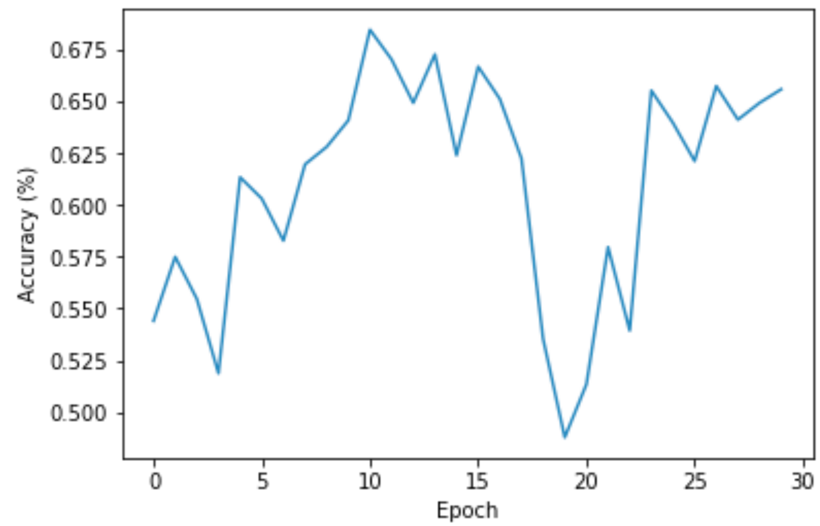
New classification

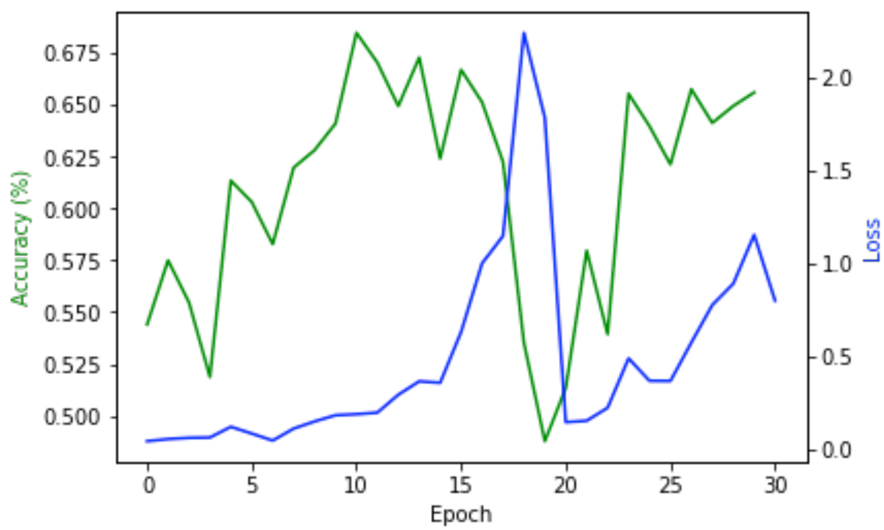


- Run again on the CNN
    - Not training (weights)
- Support Vector Machines
    - Last layer of the CNN (classifier) as trained
    - vs. validation or test set
    - labels

Results







Results

- Accuracy on test set: 67%
- Confusion matrix

Class	1	2	3
1	62.8%	37.2%	0%
2	26.0%	74.0%	0%
3	58.2%	41.8%	0%

Test set on Epoch 12

Class	1	2	3	n%	N%
1	826	490	0	42.8%	43.3%
2	433	1233	0	54.2%	54.2%
3	53	38	0	2.96%	2.52%

n: 3073

Results

- Timing
  - ~10.5 hours (only training the network)
  - ~1 hour (learning and extracting features)
  - 3073 images —> ~12 seconds per image

Conclusions

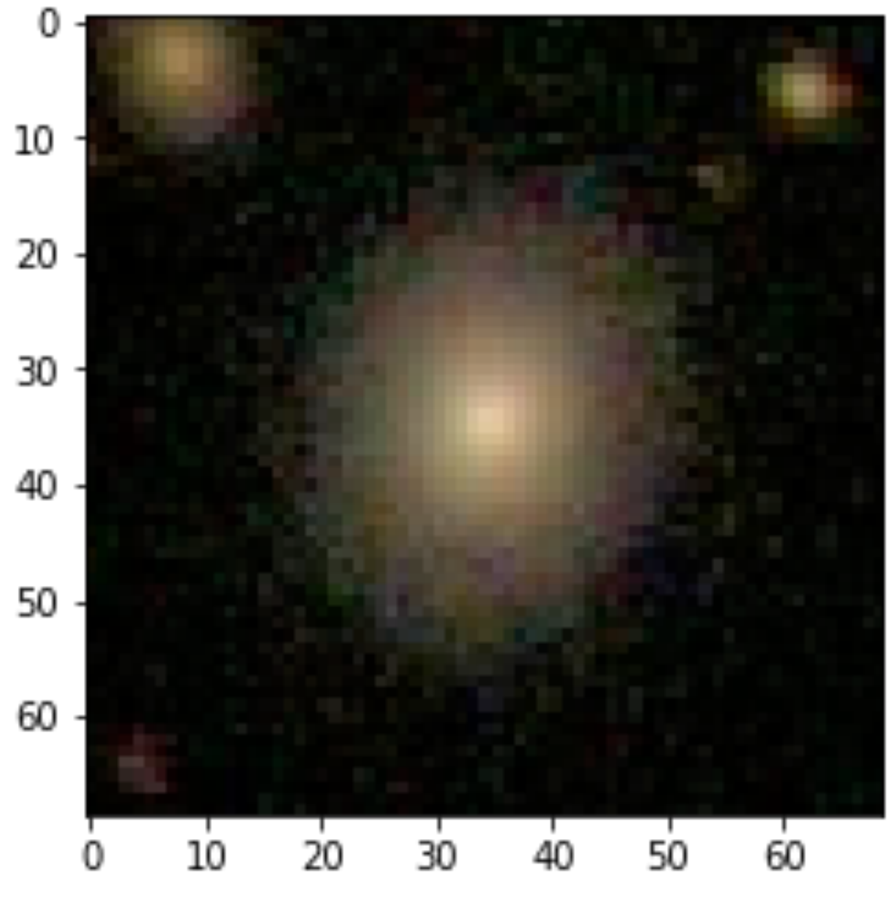
- CNN vs SVM
- Promising approach
- Contribution to galaxy classification and to learning by example

https://github.com/iled/galazyxoo

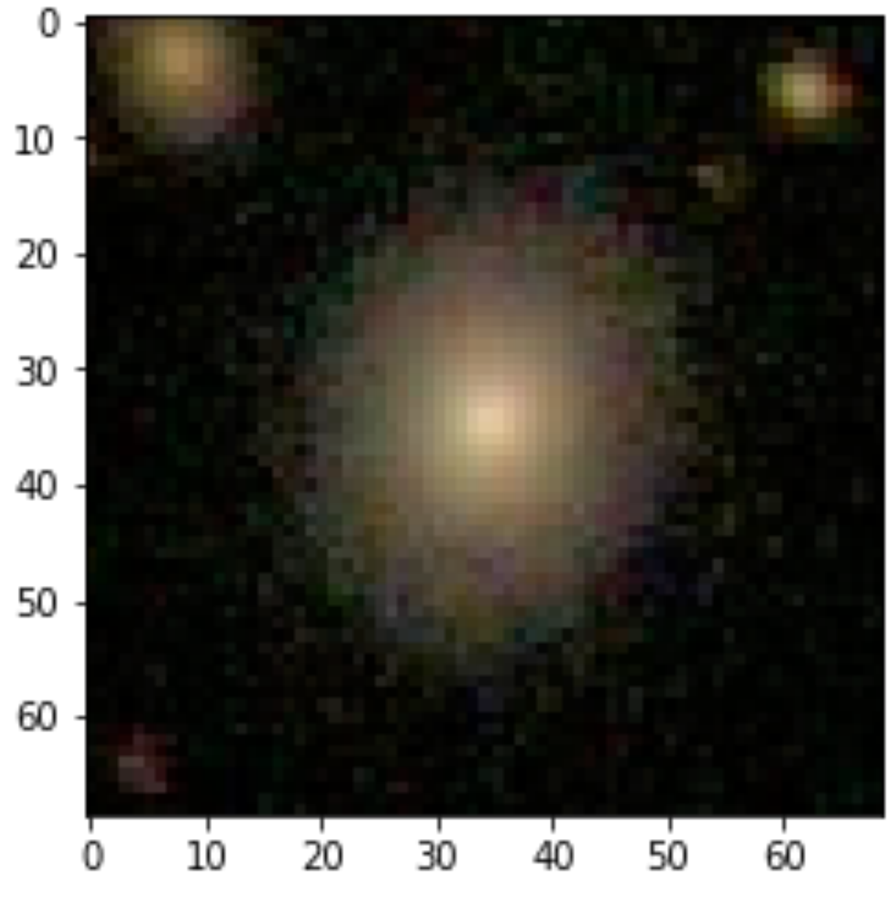
Future work

- Compare with other approaches
- Improve timing
- Regression
- Different methods of classification on top of CNN

Color perturbation



Color perturbation



Color perturbation