Artificial Intelligence on the Final Frontier: Characterizing Galaxy Morphology using Computer Vision

Abraham Botros
SUNet ID: abotros
abotros@stanford.edu
Winter 2015, CS231A Final Project Report, Stanford University

1 Introduction

For ages, mankind has looked towards the stars and wondered what lies out there in the great beyond. Now, with recent advancements in technology, we can use powerful telescopes to observe celestial objects over unfathomable distances. In particular, by peering beyond the limits of our own cosmic neighborhood, the Milky Way, we can even begin to visualize other galaxies and learn more about just what is out there in the universe.

In this project, we examine data provided as part of the Galaxy Zoo 'Galaxy Challenge' on Kaggle [1]. This challenge's dataset includes thousands of images of galaxies as obtained by a variety of sky surveys, such as the Sloan Digital Sky Survey [2, 3]. Using crowd-sourced labels (from [2]), the training images are accompanied by probability distributions for galaxy morphology characteristics (such as smooth, cigar-shaped, etc.) according to the Galaxy Zoo 2 decision tree [4] (see Figure 2) and evaluated by human Galaxy Zoo participants. Given training images to learn a mapping from raw image files to probability distributions for galaxy morphology characteristics, the goal is to output accurate probability distributions given raw image files at test time (see Section 3).

Given the incredibly large amounts of data gathered continuously by space telescopes, automated classification is essentially necessary for the scientific community to even begin to process and analyze all this information. In particular, with countless galaxies out there for us to discover and analyze (more than 60,000 different galaxies in just the training set for this Kaggle competition), and with the inherent noisiness in such observations and widespread differences in the galaxies themselves, machine learning using computer vision techniques should provide extremely useful in this and related applications. Furthermore, in general, learning more and more about the galaxies that exist beyond our own will help us to better understand the types of galaxies that exist, how these types are distributed, the physics behind various galaxy morphologies, the formation of new and existing galaxies, our own home in the universe, and ultimately, the universe as a whole.

2 Previous Work

2.1 Overview

In general, the scientific space community has been identifying and classifying galaxy morphology for years. Typically, such characterization is modeled as a classification problem, with only a very small number of possible classes for a given galaxy image, such as elliptical, spiral, edge-on, etc. [6, 7, 8]. While the domain is clearly similar, the problem here is more difficult, as we seek to discern much more fine-grained details for each galaxy image (instead of just predicting/classifying perhaps the highest-level relevant information possible for each galaxy, we want to actually describe its parts as a regression problem).

To the best of the author's knowledge, the crowd-sourced Galaxy Zoo project was the first major undertaking for characterization of fine-grained galaxy features on truly large-scale (though not exactly automated). The resulting Kaggle competition here directly extends from the Galaxy Zoo crowd-sourcing project, and seems to be the first instance of very large-scale, automated, fine-grained galaxy morphology characterization.

To that extent, the main/only particularly-relevant set of literature comes from this very Kaggle competition. Since this Kaggle competition ended about a year ago, some solutions were discussed and posted (particularly for winning solutions) on the Kaggle competition forum. In particular, the winning solution (described at [5]) implemented a very thorough and complex deep convolutional neural network (CNN) with about 42 million parameters and that took 67 hours to train on a powerful workstation network. The majority of other relatively successful solutions appear to have used similar CNN approaches; there were very few public mentions of attempts to use more conventional computer vision methods (such as Bag of Visual Words, HOG/SIFT/SURF descriptors, etc.).

2.2 Differences

The majority of Kaggle competitors spent months on this project and brought in apparently extensive knowledge of neural networks, deep learning, and CNN's in particular. The current work seeks to see how more traditional and simplistic computer vision methods can go in this paradigm, with only a month or so of part-time time to dedicate to this. In addition, we do some basic exploration into naïveCNN implementation using a relatively shallow network. Primarily, this project work is to further the author's hands-on technical experience with various computer vision methods, and is mostly an implementation-and-adaptation work.

3 Technical Details

3.1 Summary

In summary, given thousands of training and testing galaxy images, we do the following (with some looseness in terms):

- Apply the Bag of Visual Words (hereafter, BoVW) model using standard computer vision descriptors.
  - Preprocess the images, including cropping and smoothing.
  - Extract SIFT, dense SIFT, or SURF descriptors, along with some summarizing color information.
  - Construct a BoVW model across all training images.
  - Apply standard machine learning algorithms, such as K-nearest neighbors and ridge regression, to perform regression and predict the 37 output categories.
- Use CNN's to directly compute the regression output.
  - Preprocess the images, including cropping, smoothing, and generous downsampling.
  - Construct and run a CNN taking the preprocessed image pixels directly as input, and directly outputting predictions for the 37 output categories for each image.

3.2 Data

The competition's training set consists of 61,578 JPG images of galaxies, each with a unique GalaxyId identifier. The competition's testing set consists of 79,975 images. Along with the training set, 'ground-truth' (labeled via crowd-sourcing, see Section 1) probability distributions for the classifications of each of the training images are provided in the form of a 37x1 vector. Test set ground-truth distributions are not directly available, and can only be tested against through the Kaggle online interface on no less than the entire testing set.

Figure 1 shows examples of input images in this dataset. For each training image in our dataset, we want to learn a probability distribution for that image over various morphology characteristics; for testing images, we want to predict their corresponding probability distributions using what we learned from the training set. There are 37 different morphology categories (corresponding to those shown in the Galaxy Zoo decision tree [4]; see Figure 2), and each category is assigned a floating point number between 0 and 1 inclusive.

As we can see, even though it is very convenient that all input images are nicely centered, the images in general are noisy and diverse, and often have only very fine visual differences that set them far apart on the decision tree (such as the presence of a small darker areas signifying the presence of a galactic spiral arm, for example). Images are relatively grainy, and this graininess becomes especially noisy if we look at pixels along the boundaries between the celestial bodies and the surrounding void/blackness. The noisiness of the data is further compounded by the frequent presence of additional distracting celestial bodies near the center of the image.

Lastly, we also note that we must attempt to stay invariant to rotation (scale and translation invariance are already obviated for the most part). There is no sense of direction in space; as a result, we do not care of a cigar-shaped galaxy is orientated in one direction or another.

For all these reasons, this problem becomes quite hard - our algorithms must be robust enough to handle all these noise factors, distractors, and rotational-variance cues, yet must be precise enough to accurately perform regression prediction for 37 different morphology categories.

3.3 Software

Virtually all code was implemented using Python, along with the Python OpenCV package [10] and Python scikit-learn package [11]. Convolutional neural networks were applied using Caffe [12].

3.4 Preprocessing

The first step was to do some standard preprocessing of the images. Each galaxy image was given as a color image of size 424x424 pixels. Manual investigation of the dataset showed that cropping down to the center 170x170 pixels yielded several benefits:

- All primarily important information was contained in the center 170x170 pixels for virtually all images.
- Cropping out the surroundings helped to eliminate many distractor nearby galaxies in the images, along with other distracting celestial bodies and noise.
- This yielded a huge dimensionality reduction for keeping machine learning on this dataset feasible.

Secondly, all images were smoothed with a standard Gaussian filter to eliminate some of the granularity and pixelation in the original images.

3.5 BoVW Features

3.5.1 SIFT and Dense SIFT

We used SIFT feature descriptors to extract features from each galaxy image. Both traditional SIFT [9] and dense SIFT feature descriptors were used through OpenCV, where traditional SIFT features were located using a relaxed contrast threshold of 0.02 (otherwise, very few descriptors were located in a given image, if any). See Figure 3.

In training, we randomly sample a large subset of the total descriptors extracted across all training images, and use these descriptors to train a BoVW model with N words/clusters. We then go back through the training set to assign each training image a normalized BoVW histogram according to the occurrence of its descriptors and the nearest of the N words/clusters, as is typical for BoVW models. Thus, each training image is now represented by a fixed-length vector corresponding to its BoVW histogram representation.

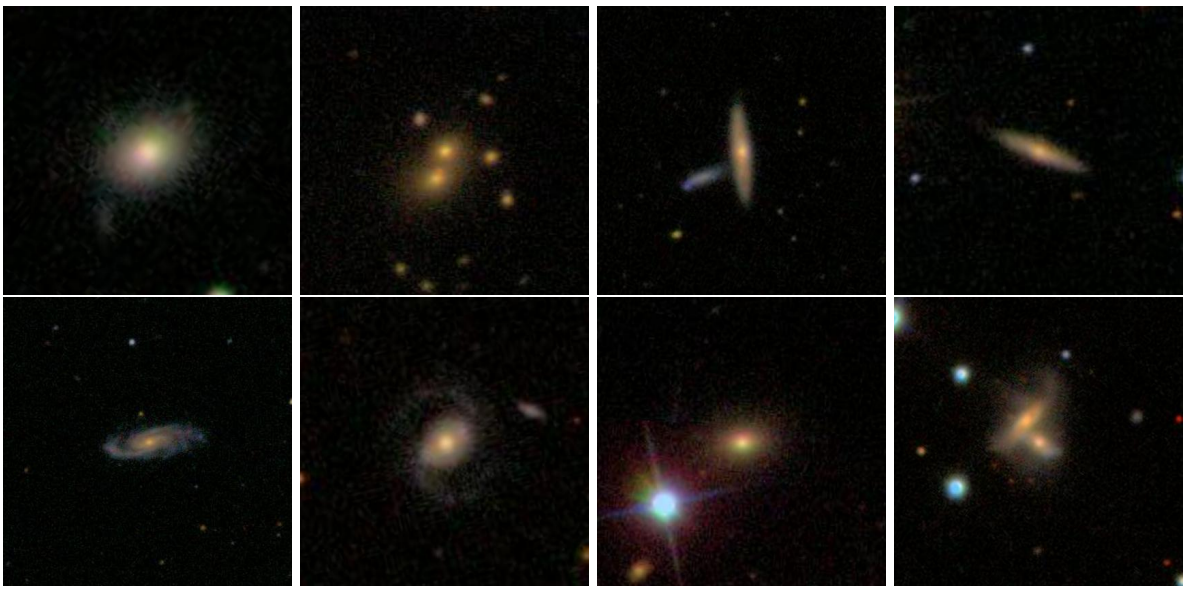


Figure 1: Several examples of galaxy images from the dataset. This set of images is fairly representative of the typical images found in this dataset as a whole.

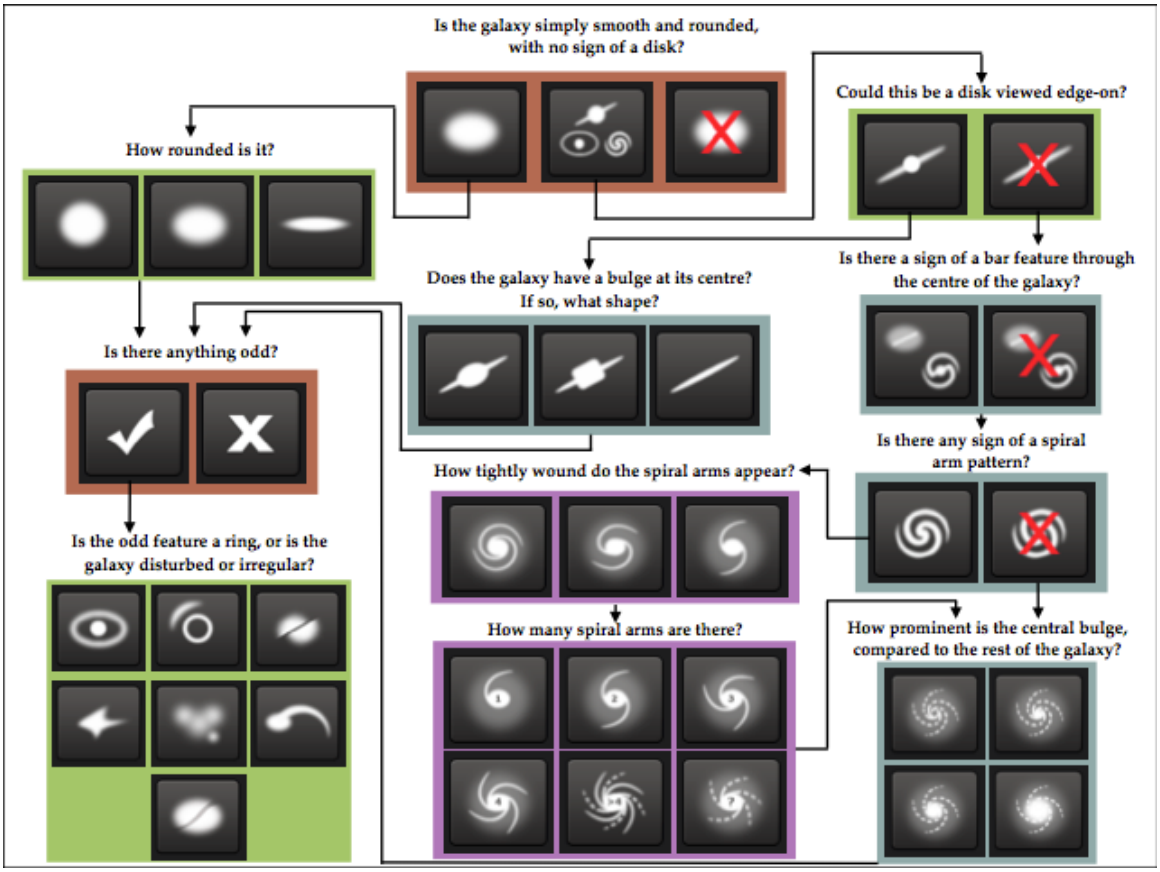
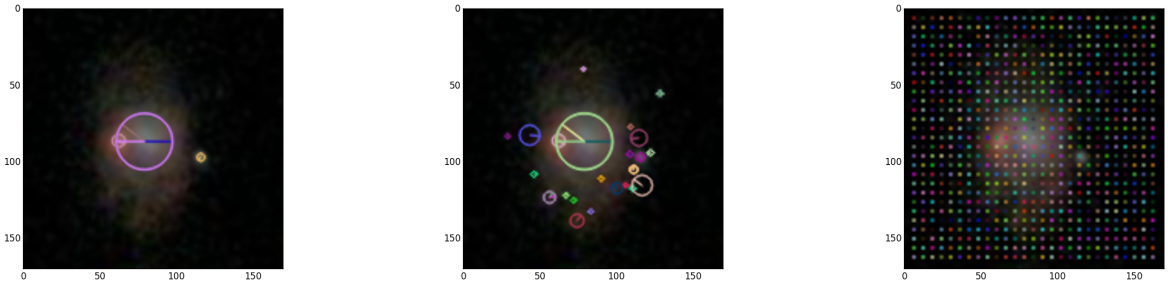


Figure 2: Galaxy Zoo decision tree. Shows a flowchart of the classification tasks given to human participants in the online Galaxy Zoo crowd-sourcing project. Probability distributions over all participants for these 37 categories are used as ground-truth in this project's algorithms and in the Kaggle competition as a whole. Tasks are color-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. See [4] for more information, including a detailed table of questions and responses for each task.



(a) Default SIFT **(b)** Relaxed-contrast SIFT **(c)** Dense SIFT

Figure 3: Examples of various SIFT feature descriptors detected in a typical image. Note that the image is relatively non-salient, which is common in this dataset. As we can see, the default SIFT detector/descriptor algorithm picks up very few descriptors in the image, and does not really capture much of the interesting information. The relaxed-contrast SIFT does much better with this, fitting descriptors where we might hope. On the other hand, dense SIFT simply calculates descriptors at fixed grid points in the image.

3.5.2 Rotational Invariance

To begin to get at the rotational invariance critical for robust learning, regression, and performance in general, we preserve some rotational location information when calculating the BoVW histogram above. In particular, instead of using the traditional spatial pyramid approach which segments the image into box compartments, we compartmentalize descriptors in the image according to their Euclidean distance from the center (similar to the circular spatial context descriptor approach we saw in class). In essence, this results in creating "radii zones", or concentric circles at evenly spaced radii/distances from the center, demarcating distances of image points from the center of the image. As is typical in spatial pyramid and spatial context descriptor approaches, outer/larger-scale regions accumulate counts from inner/smaller-scale regions contained within them. See Figure 4.

In doing this, all descriptors found at a certain distance from the center, regardless of rotation angle/orientation from said center, are pushed into the same histogram. This gives our BoVW model some decent rotational invariance, which will be important for learning later on.

3.5.3 Concatenation

The best models from training/cross-validation used 1000 clusters/words and 5 spatial radii zones (as shown in Figure 4). This resulted in a 5000-dimensional vector for each image's BoVW features. This was then concatenated with the 13-dimensional color feature vector (described in Section 3.6) to create a 5013-dimensional feature vector in total for each image for learning and prediction.

3.6 Color

The best benchmark given in the Kaggle competition was the "central pixel benchmark", which clustered training galaxies according to the color in the center of the image and assigns associated probability values to like-colored images in the test set; this achieved a respectful root-mean-squared error (RMSE) of 0.16194 on the testing set (see Section 4). This indicated that color was relatively informative of the galaxy morphology in general; this makes sense, considering color may be indicative of characteristics involved in that galaxy's composition, formation, and therefore morphology.

As a result, we extracted some summary color/HSV information from each image, in addition to the BoVW features above.[1] In particular, the color features include two parts:

- A 3x1 vector of the raw HSV values of the center pixel of the image.
- A 10x1 normalized histogram of hue values across the entire (preprocessed/cropped image), where all hue values are accordingly binned into 10 histogram bins and then normalized.

As discussed in Section 3.5.3, this 13x1 vector was then concatenated with the BoVW feature vector for learning and prediction.

3.7 CNN Features

3.7.1 Additional Preprocessing

In addition to the preprocessing already described in Section , we downsample each image twice (down to 1/4-th size, or down from 170x170x3 to 43x43x3. This allows training the CNN network to be feasible given our very limited computational resources.

3.7.2 Rotational Invariance

Lastly, to give the CNN network some rotational invariance, we replicate each training image an additional 3 times for each 90° rotation. Thus, for each training image, we pass in the original, unrotated image; we then rotate the original image by 90°, 180°, and 270° to generate three new rotated images, which we then also pass into the network with the same labels as the original (the same 37 ground-truth output categories).

Ideally, this would instill in the CNN some rotational invariance, since it must learn that (coarse) rotations of a given image lead to the same desired output. This approach was

[1] Note that this only applies to the BoVW + non-CNN approach, as the CNN approach directly uses pixel values, including pixel color.

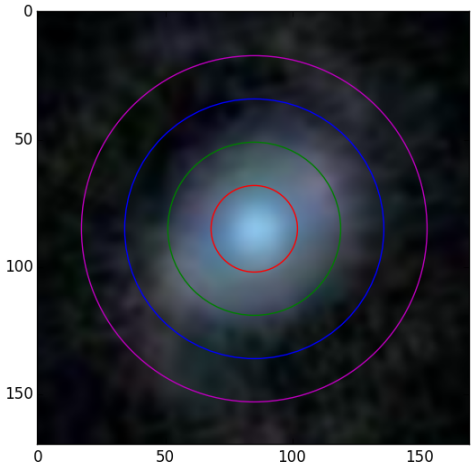


Figure 4: Illustrative example of the spatial radii zones described in Section 3.5.2.

similar to the rotation-relevant preprocessing in the winning Kaggle solution (again, described at [5]).

As expected, test images were not rotated in the same way, and were simply tested on as-is by computing a single forward pass on the network.

3.8 Learning

3.8.1 BoVW-centric Approach

For the BoVW-centric pipeline, where each training image is represented by a 5013-dimensional feature vector representing BoVW features and color features, we input our entire training set for learning (or 0.7 of it for training and 0.3 for cross-validation if doing training/cross-validation).

To perform learning and prediction, we use the Python scikit-learn package to run K-nearest neighbors regression, ridge regression, and support vector regression (SVR). Cross-validation (as mentioned above) was used to choose hyperparameters for testing. See Section 4 for results.

Given a test image, we extract the same BoVW and color features as we did in training. We then make a prediction based on the nature of the algorithm used:

- In K-nearest neighbors regression, given our feature vector for our test image, we simply search the full, in-memory dataset for the K most similar training examples in this BoVW+color feature space, then output a weighted average of the 37 ground-truth output category values among those K neighbors. Thus, we easily can generalize from regression of one output value to regression of an arbitrary number of output values.
- For ridge regression and support vector regression, this is a bit more complicated, as these models typically only output a single value. To circumvent this, we simply train 37 different instances of each model (one for each output category). At test time, given a test image, we run this single test image through each of the 37 different trained models to get 37 different outputs.

3.8.2 CNN Approach

The author had essentially no experience with CNN's before this project, so much of the exploration into CNN's in this work was relatively trial-and-error. Using the open-source Caffe CNN library, we constructed and experimented with various network structures. In the end, we settled on training and testing using the network structure defined by Krizhevsky et al. in their well-known work in the ImageNet classification challenge [13] (the network structure was essentially integrated into Caffe as an example, and needed only modification of input/output; see [14]); we also train and test on a custom, modified, truncated network with some components and structure similar to the Krizhevsky network, the famous LeNet/MNIST network (see [15]), and the network used in the winning solution (see [5]), though with some liberal experimentation and butchering for personal learning and for making the network more feasible for running in reasonable time.

See the accompanying references and project code for the detailed structure of the networks used.

4 Experiments

4.1 Evaluation Metric

Root-mean-squared error (RMSE) was used as the evaluation metric for this project (in line with the evaluation used in the Kaggle competition). This error over the entire training/testing set is computed as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - a_i)^2} \tag{1}$$

Where N is the number of galaxies times the total number of responses (i.e., 37 for each galaxy), p_i is the predicted value,

and a_i is the actual value.

4.2 Results

Results for all major test conditions are shown in Table 1.

Unfortunately, the SVR algorithms simply took too long to run on even a truncated (but still reasonably-sized) slice of the dataset, and was left out of certain permutations because of this. From full tests it did participate in, it appeared to have similar performance to ridge regression, not adding anything better.

Analysis for the experimental results can be found in Section 5.1.

5 Conclusion

5.1 Analysis

5.1.1 BoVW+Color Approach

In general, performance achieved was reasonable, but not spectacular. It seems the BoVW+color approach hit a wall around the 0.15-0.17 test RMSE area, even with a more expressive BoVW vocabulary (1000 over 100 words) and more raw features extracted (dense SIFT vs vanilla SIFT). All machine learning algorithms performed similarly on top of this BoVW+color approach.

A more simplistic approach of simply clustering based on central colors (the best benchmark given) and using a nearest-neighbors learning algorithm on top of this did comparably to the full BoVW+color approach described in this project.

It may be that feature descriptors such as SIFT are more useful for more artificial, descriptive objects (such as those found in more normal, every-day scenes in datasets such as ImageNet, etc.), and relatively different, more fluid and deformable paradigms (such as the galaxy characterization here) are less suited for such descriptors.

5.1.2 CNN Approach

Neither CNN model was able to run for too many iterations (stopped at 1000-2000 iterations) due to the time of computation, meaning they could have done better given more training time and computational resources (in particular, the better-formed ImageNet network).

However, for reasons similar to those described in Section 5.1.1 above, the particular network structure fine-tuned for the ImageNet dataset may also be not particularly suitable for the task at hand, as described by the winning solution at [5]. A different CNN structure might have been more useful here due to the nature of this dataset, such as the one carefully and expertly created in the winning solution.

The experimental CNN, while giving the author good exposure to various minutiae of CNN design and performance, did not perform exceptionally well, though it did run substantially quicker than the full ImageNet CNN. Further tweaking could likely increase performance here.

As a side note, it may also be that bugs still reside in the very-new Caffe library. Numerous bugs still currently plague the library, especially in the command-line and Python interfaces. In particular, the regression paradigm described here (where training images have a corresponding label of a 37-dimensional vector) may be particularly erroneous (numerous issues such as the one at [16] exist in the Caffe repository that are unresolved, especially if using networks that differ even slightly from the couple of given example networks). While the author certainly could use more experience in interfacing with the Caffe library and these problems may well be the fault of the author, numerous instances of other researchers running into similar problems in the last few months using Caffe are easily discoverable on the web. Regardless of the source of the bugs (whether caused by the author or the Caffe code itself), resolution of said bugs and issues could significantly improve Caffe CNN performance in this project in the future.

5.2 Future Work

5.2.1 BoVW+Color Approach

Several further augmentations might be able to boost performance for the BoVW+color approach. In particular:

- **Increasing the BoVW vocabulary size.** Due to the tremendous size of the dataset and huge variations between images even of similar galaxies, it may well be that further increasing the BoVW vocabulary size may be able to better capture the full range of differences and variations in the dataset.
- **PCA and other dimensionality reduction.** Conversely, reducing dimensionality with methods such as PCA may also provide better machine learning performance given a reasonable BoVW+color descriptor vector for each image. This would certainly be an important priority going forward.
- **Better locality and rotational invariance.** Augmenting the training dataset for the BoVW approach using image rotations similar to the approach described for CNN input above (see Section 3.7.2) may also have better given this approach robustness to rotation and exact locality of features.

5.2.2 CNN Approach

Likewise, several improvements could likely help the CNN approach:

- **More CNN domain expertise.** Better domain expertise with crafting and debugging complex networks would almost certainly drastically improve CNN performance here, as evidenced by the winning solution at [5].
- **More computational power.** Running with more iterations, more/deeper layers, and with more preprocessing (more input rotations, less downsampling, more memory, etc.) would likely improve CNN performance.

Model	Descriptor	Number of BoVW Words	Train RMSE	Test RMSE
All zeros benchmark	-	-	-	0.27210
All ones benchmark	-	-	-	0.89448
Central pixel benchmark	-	-	-	0.16235
-	-	-	-	-
kNN	SIFT	100	0.13390	0.16409
kNN	SIFT	1000	0.12413	0.17404
kNN	Dense SIFT	100	0.12996	0.15952
kNN	Dense SIFT	1000	0.01603	0.16993
Ridge	SIFT	100	0.15823	0.15819
Ridge	SIFT	1000	0.16923	0.16907
Ridge	Dense SIFT	100	0.16248	0.16235
Ridge	Dense SIFT	1000	0.15673	0.15655
SVR	Dense SIFT	1000	0.17722	0.17802
-	-	-	-	-
ImageNet CNN	-	-	0.16980	0.15943
Experimental CNN	-	-	0.185906	0.18601

Table 1: All experimental results. The first section indicates given Kaggle benchmarks; the middle section shows the main BoVW+color approaches; and the last section shows performance for the CNN's tested.

- **Category-specific decomposition.** As in the BoVW approach, it might also be helpful to train a CNN for each output category.

5.3 Closing

Overall, we explore various different computer vision approaches to this fine-grained galaxy morphology characterization problem, including Bag of Visual Word and convolutional neural network approaches. All methods performed reasonably, though there is still certainly room for improvement. Limitations on work time, computational resources, and prior expertise hindered achieving record-breaking performance, but this project still proved enormously useful to the author's own experience with computer vision.

In general, this was an exciting domain to work in within computer vision, and the author certainly hopes to one day be able to much better apply computer vision and machine learning techniques to help mankind better understand our origins and our place in the stars.

6 References

[1] See https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge.

[2] See http://www.galaxyzoo.org/.

[3] See http://www.sdss.org/.

[4] See https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge/details/the-galaxy-zoo-decision-tree.

[5] See http://benanne.github.io/2014/04/05/galaxy-zoo.html.

[6] Goderya, Shaukat N., and Shawn M. Lolling. "Morphological Classification of Galaxies Using Computer Vision and Artificial Neural Networks: A Computational Scheme." *Astrophysics and Space Science* 03-2002 279.4 (2002): 377-87. *SpringerLink*. Web: http://link.springer.com/article/10.1023/A%3A1015193432240.

[7] Shamir, Lior. "Automatic Morphological Classification of Galaxy Images." *Mon. Not. R. Astron. Soc.* 399.3 (2009): 1367-1372. *PMC*. Web: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2808694/.

[8] De la Calleja, Jorge, and Olac Fuentes. "Machine Learning and Image Analysis for Morphological Galaxy Classification." *Mon. Not. R. Astron. Soc.* (2003). Web: http://www.cs.utep.edu/ofuentes/mnras5.pdf.

[9] Lowe, David G. "Object recognition from local scale-invariant features." *Proceedings of the International Conference on Computer Vision.* 2 (1999): 1150-1157.

[10] See http://opencv.org/.

[11] See http://scikit-learn.org/stable/index.html.

[12] See http://caffe.berkeleyvision.org/.

[13] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in Neural Information Processing Systems* 25 (2012): 1097-1105.

[14] See http://caffe.berkeleyvision.org/gathered/examples/imagenet.html.

[15] See http://caffe.berkeleyvision.org/gathered/examples/mnist.html.

[16] See https://github.com/BVLC/caffe/issues/1396.