ROTATION INVARIANT FEATURE EXTRACTION IN THE CLASSIFICATION OF GALAXY MORPHOLOGIES

S. REITSMA

SUPERVISOR:
PROF. DR. T.M. HESKES

SECONDARY SUPERVISOR:
DR. L.G. VUURPIJL

Bachelor of Science thesis
Artificial Intelligence
Faculty of Social Sciences
Radboud University Nijmegen

4 July 2014

ABSTRACT

The Galaxy Zoo project is a crowdsourcing platform to classify the morphology of galaxies into different categories. Recently, the project set out a challenge to automatically predict these crowdsourced classifications using machine learning techniques. In this thesis, one of these machine learning techniques is explored and modified. This technique, designed by Coates et al. [4], works by learning features in an unsupervised manner using k-means. These features are rotation sensitive, but since there is no up or down in space, the system would ideally work rotation invariantly. Therefore, the system was modified to account for rotation sensitivity in an efficient manner by changing the distance metric that is used by the k-means algoritm. Results show that this improves the performance significantly by more than 5%: the regular method achieves a root mean squared error of 0.10789 while the rotation invariant method achieves a score of 0.10256.

CONTENTS

1 FROM HARD LABOR TO AUTOMATION

1.1 STELLAR CLASSIFICATION

> *Classifying the stars has helped materially in all studies of the structure of the universe. — Annie Jump Cannon*

In the final years of the 19th century, a woman named Annie Jump Cannon joined a team at Harvard led by Edward C. Pickering to create a catalogue of the brightness of the stars. During Cannon's time at Harvard, she invented a stellar classification system that is still used to this date[1]. The system was based on emission spectra, which show the frequencies of emitted radiation due to electrons transitioning from one energy state to the other. These transitions are characteristic for a certain element. By using the emission spectra, Cannon was able to determine the ratio of elements in a star. Later, these spectra were determined to be related to stellar temperature. In her lifetime, Cannon manually classified approximately 500,000 stars.

1.2 FAST-FORWARD TO THE START OF THE 21ST CENTURY

In the year 2000 the Astrophysical Research Consortium started the Sloan Digital Sky Survey project. In this survey, the redshift of distant celestial objects is measured to estimate their distance from Earth and ultimately build a model of the visible universe [10]. The survey data now comprises millions of objects and more data is added every day. This incoming stream of data is far greater than could ever be processed manually by an individual.

1.3 GALAXY ZOO

In July 2007, the Galaxy Zoo project saw the light of day. Initially, the goal of this project was to classify the morphology of galaxies in three categories: elliptical galaxies, merged galaxies and spiral galaxies. The first instalment of the project used the SDSS[2] data set, which back then consisted of approximately a million galaxies. The classification of these morphologies was not automatized. Instead, visitors of the Galaxy Zoo website could vote on the morphology class given a photograph of a galaxy. During the first year of the project, fifty million classifications were received for almost a million galaxies.

1 Aptly named the Harvard spectral classification system
2 Sloan Digital Sky Survey

The current (fourth) version of the project uses data from more than just the SDSS [9]. The CANDELS survey uses the Hubble telescope to photograph the most distant galaxies and additionally the UKIDSS[3] provides information on the age of stars, thus revealing the more intricate structures of the inner parts of galaxies.

(a) An elliptical galaxy (NGC6782).

(b) A spiral galaxy (M51).

(c) Two merging galaxies (NGC4676 A&B).

Figure 1: An example of each of the three morphology classes in the initial version of the Galaxy Zoo project.

1.4 AUTOMATION

While crowd sourcing the morphology classifications of galaxies is a great idea and much faster than having researchers classify the galaxies manually, the amount of data is just too vast to keep up with. Luckily, technological advancements have made it possible to automatically classify these galaxies, using machine learning techniques. In this thesis one of these classification techniques is explored, designed by Coates et al. [4], and it is adapted to make it more suitable for galaxy morphology classification.

3 UKIRT (United Kingdom Infrared Telescope) Infrared Deep Sky Survey

1.5 KAGGLE COMPETITION

The Galaxy Zoo team and Winton Capital teamed up with Kaggle, a machine learning competition platform. They created a challenge where participants are to automatically classify galaxy morphologies using high resolution photographs. The main goal of the challenge is to see whether it is possible to devise a way to classify galaxy morphologies automatically, where the classifications are as close as possible to the crowd sourced classifications obtained from the Galaxy Zoo website. The competition ran from December 20, 2013 until April 4, 2014. The winner of the competition was able to achieve a score[4] of 0.07492.

1.6 RESEARCH

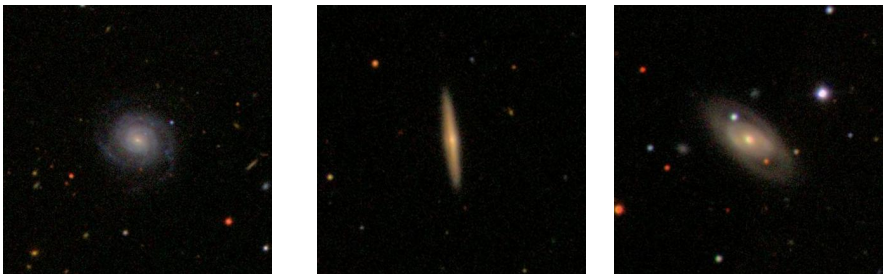
The classification strategy designed by Coates et al. [4] relies on unsupervised learning to extract features from images. The extraction of features is however *rotation sensitive*. While this is useful in most images, as an object that is rotated may have a different morphology, it does not make sense to use a rotation sensitive system in recognizing galaxy morphologies. All galaxies are photographed from Earth, and since there is no up or down in space, the rotation of the photograph is irrelevant for the morphology of the galaxy. In the next chapter, some background information is given and an adaptation of the strategy is proposed to make the system *rotation invariant*. Note that the features that are extracted are still sensitive to rotation, but the algorithm works around this by modifying the distance metric. This is explained in detail in Chapter 4. In the final chapters, the results of the rotation invariant system and the regular system are compared.

4 RMSE, lower is better

2 DATA SET

2.1 IMAGES

Galaxy Zoo has prepared a set of training and test images. All these images come from the second version of the project, as part of the SDSS catalog. This catalog contains many images that were photographed in a region known as Stripe 82, which has been repeatedly imaged over the years [1]. In this region of space, the SDSS has taken multiple images of various galaxies. The images are all 424x424 3-byte (RGB) pixels, which amounts to a total of 539328 bytes per image. In total, there are 61578 images in the training set and 79975 in the test set. This amounts to more than 71 gigabytes of data when uncompressed.



(a) Galaxy #100765. (b) Galaxy #101355. (c) Galaxy #801467.

Figure 2: An example of three of the images in the training set.

2.2 REGRESSANDS

The images would need to be categorized into several classes. However, instead of merely predicting the morphology class, the classifier needs to give a distribution of votes for each question, as explained in further detail below. This comes down to a total of 37 values that need to be predicted. These values correspond to the fraction of votes on that particular class for the image, as gathered by the Galaxy Zoo project. This creates a regression problem instead of a classification problem.

2.2.1 *Decision tree*

The 37 values are not all on the same level. There is a hierarchy woven into them, as illustrated in Figure 3.

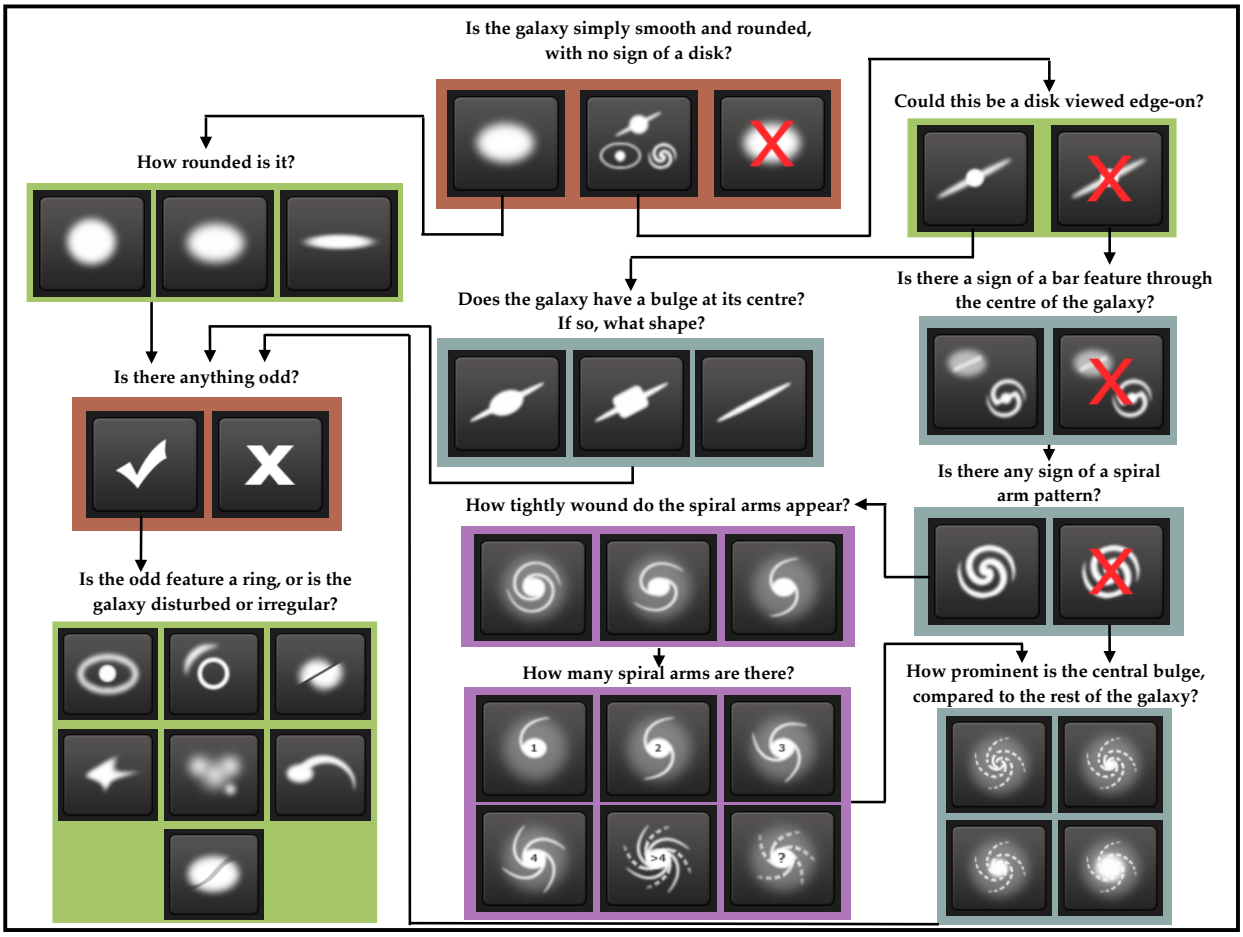


Figure 3: The galaxy zoo decision tree [9].

In total, the decision tree consists of 11 questions, with each question having between 2 and 7 possible responses. The questions are the following:

1. Is the object a smooth galaxy, a galaxy with features/disk or a star/artifact? 3 responses
2. Can the galaxy be viewed edge-on? 2 responses
3. Is there a bar? 2 responses
4. Is there a spiral pattern? 2 responses
5. How prominent is the central bulge? 4 responses
6. Is there anything "odd" about the galaxy? 2 responses
7. How round is the smooth galaxy? 3 responses
8. What is the odd feature? 7 responses
9. What shape is the bulge in the edge-on galaxy? 3 responses

10. How tightly wound are the spiral arms? 3 responses

11. How many spiral arms are there? 6 responses

All galaxies in the image sets were classified by humans and the output consists of a distribution of those human classifications. As a result, the probability at each question will sum up to 1. However, the probabilities are weighted. For the first question, this has no effect, but for subsequent questions, probabilities are multiplied by the value which led to that question. As an example, consider the following scenario: for the first question a probability distribution of $\{.8, .15, .05\}$ was found. For 80% of the users, the next question was Q7 (see the tree in Figure 3). If the distribution of responses for Q7 was $\{.5, .25, .25\}$, then the weighted distribution would be $\{.5 \times .8, .25 \times .8, .25 \times .8\}$. An exception to this rule is question 6, which is normalized to sum up to 1.

2.2.2 *An example*

An example of the probability distribution for galaxy #100765 (as shown in Figure 2a) is shown in Table 1.

Question 1:	Smooth 0.069821	Features/disk **0.928216**	Star/artifact 0.001962	
Question 2:	Yes 0	No **0.928216**		
Question 3:	Yes 0.108152015	No **0.820063985**		
Question 4:	Yes **0.928216**	No 0		
Question 5:	No bulge 0	Noticeable **0.637079195**	Obvious 0.291136805	Dominant 0
Question 6:	Yes 0.031074	No **0.968926**		
Question 7:	Completely 0.031908197	In between **0.037912803**	Cigar shaped 0	
Question 8:	Ring 0 Other 0	Lens or arc 0 Merger 0	Disturbed 0 Dust lane 0	Irregular **0.031074**
Question 9:	Rounded 0	Boxy 0	No bulge 0	
Question 10:	Tight **0.568020853**	Medium 0.360195147	Loose 0	
Question 11:	1 0 More than 4 0	2 **0.599247896** Can't tell 0.194647823	3 0.049749593	4 0.084570688

Table 1: Probability distribution for galaxy #100765 as shown in Figure 2a.

3 UNSUPERVISED LEARNING

3.1 PIPELINE

The classification technique described and implemented in this thesis is based primarily on the work of Coates et al. [4]. The technique consists of three phases. In the first phase, patches, randomly extracted from the training images, are fed into a k-means clustering algorithm. The centroids that result from this algorithm can be used as feature representations. In the second phase, features are extracted from the training images using the previously learned centroids. In the third phase, a classifier (or regressor) predicts the labels (or values) given the feature vectors as outputted by the feature extraction phase. A graphical overview of the pipeline is shown in Figure 4 and Figure 5. In the following sections, the phases are described in more detail.

3.2 SIZE REDUCTION

Because working with such a huge data set is computationally expensive, its size has to be reduced considerably. Input images are first cropped to a size of 150x150 pixels. This greatly reduces the size of the image, but does not necessarily signify a loss of information. This is due to the fact that galaxies are already centered, resulting in the outer border of the image to contain primarily blackness and noise.

After cropping, the images are resized to 15x15 pixels. This step does reduce the overall accuracy of the system, but is required to reduce the input size to manageable proportions. With more computational power it would be interesting to see the effect of decreasing the resize factor.

Color data is kept at all times since a galaxy's morphology is related to its temperature, which is expressed as different colors [7]. The stars in the arms of spiral galaxies are relatively young, which results in a blueish color. However, old stars in the central bulge of a galaxy are yellow-orange. This color information could help identify the bulge and differentiate between elliptical and spiral galaxies.

3.3 CLUSTERING

From these images, square patches are randomly extracted. The patch size has to be large enough to represent an image feature, but should also not be too large, since this will result in multiple features being represented in a single patch, which increases the amount of required

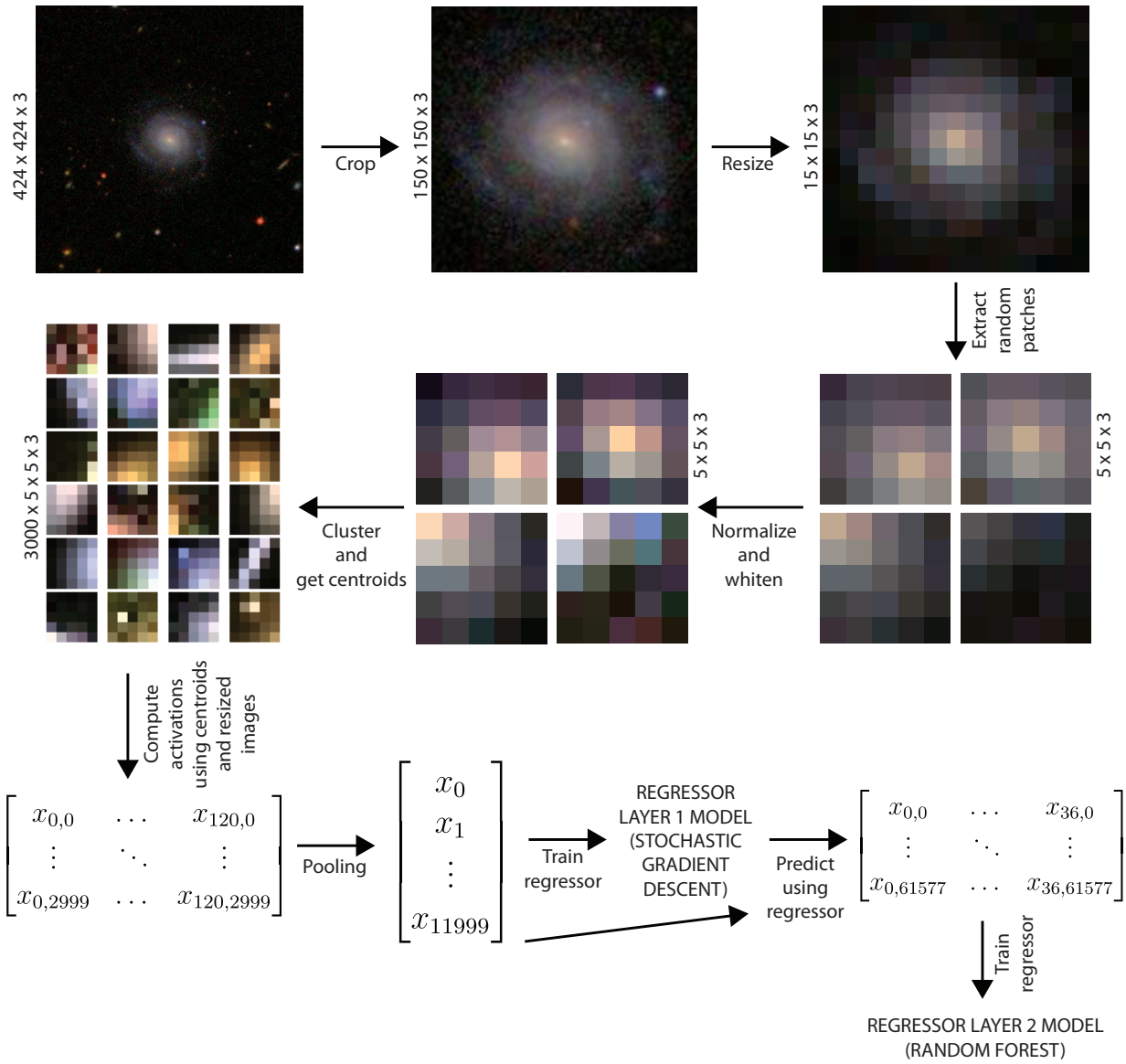


Figure 4: A graphical overview of the training pipeline.

centroids. Larger patches also result in a longer running time. Coates et al. [4] recommend using a patch size of 6x6 to 8x8 pixels when the input image size is 32x32 pixels. Since the images in this data set are over four times smaller after size reduction, a smaller patch size was used. Looking at the ratio between patch size and image size as suggested by Coates et al. [4], this would result in a patch size of 3x3 pixels for the Galaxy Zoo data set. However, since this proved to be too small to represent any features, a patch size of 5x5 was chosen.

The randomly extracted patches are normalized and clustered using k-means. An optional optimization step is to whiten the extracted

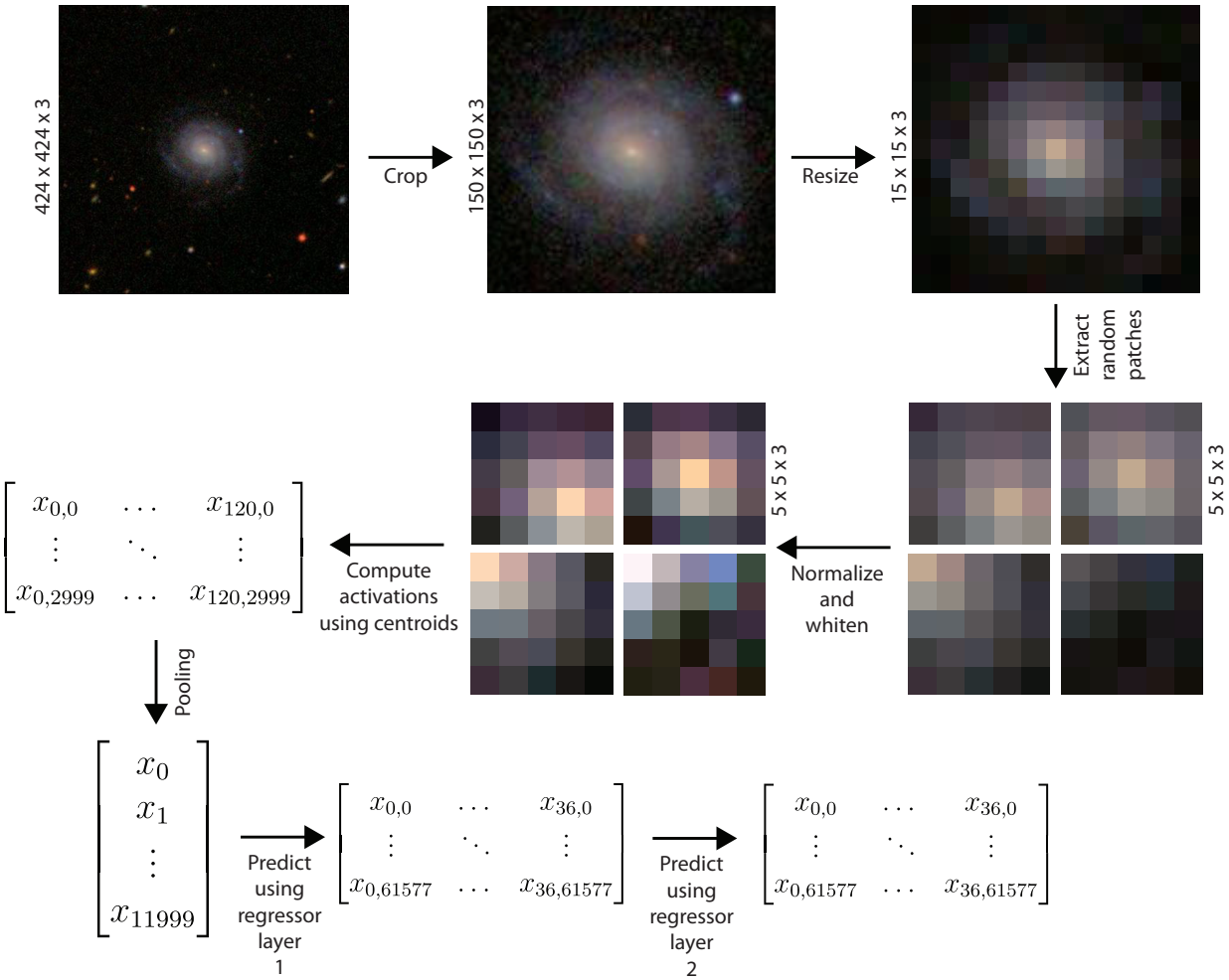


Figure 5: A graphical overview of the testing pipeline.

patches to reduce interpixel correlation. An explanation of whitening can be found in Section 3.6.

Coates et al. [4] obtained optimal results by training 1600 centroids using k-means. However, in order to account for rotation and scale variance, this value was increased to 3000 centroids. A graphical representation of the trained centroids can be seen in Figure 6.

3.4 FEATURE EXTRACTION

After obtaining the feature representations in the form of centroids, features can be extracted from the training and test images. Coates et al. [4] offer two approaches of doing this: the *hard* method and the *soft* method.

Both methods start by extracting patches from the input image convolutionally. These patches are normalized and optionally whitened. The hard method works by creating a sparse feature matrix (amount

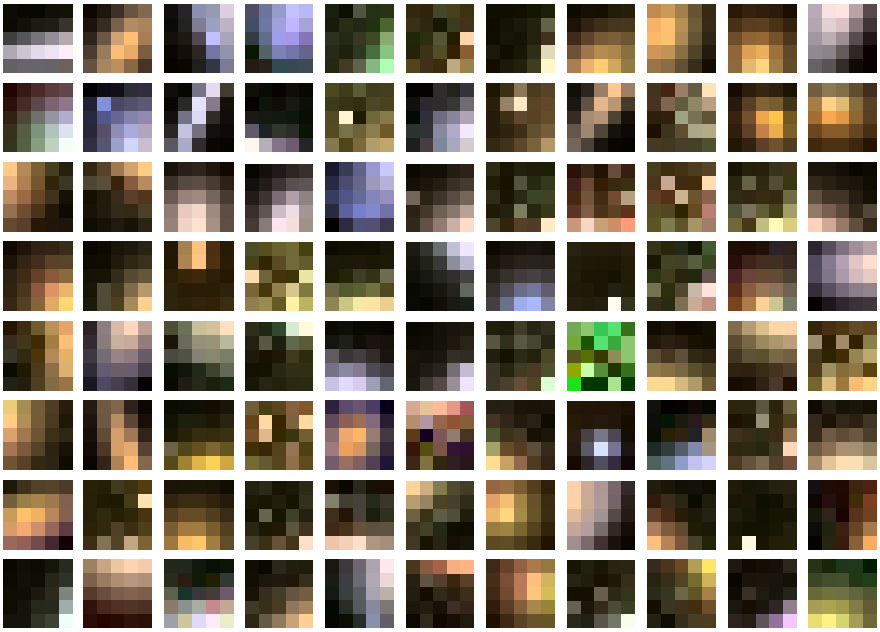


Figure 6: A random selection of 120 centroids, trained using k-means.

of patches by amount of centroids) where there is only one non-zero value for each patch at index k where the distance between the patch and the centroid $\mathbf{c}_k$ is minimal. More formally, where $\mathbf{x}$ is a patch:

$$f_k(\mathbf{x}) = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{c}_j - \mathbf{x}\|_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

However, according to Van Gemert et al. [8] this scheme is too agressive. Coates et al. [4] give an alternative: the *soft* coding scheme.

In this scheme, the resulting feature matrix is not completely sparse. The scheme function outputs 0 for features where the distance to the centroid is above average. If the distance is below average, the distance value is outputted – subtracted from the mean distance – instead of a 1, as would be the case in the hard-assignment coding scheme. Formally:

$$f_k(\mathbf{x}) = \max(0, \mu(\mathbf{z}) - z_k) \tag{2}$$

where z_k is the Euclidean distance between the patch and centroid k and $\mu(\mathbf{z})$ is the mean of the elements of $\mathbf{z}$.

The resulting feature matrix is very large. To reduce this size we can use a technique called pooling. Pooling works by summing activations in image regions. In this case, we pool over four quadrants, summing up the activations in each quadrant, resulting in a feature vector of length $4k$.

3.5 CLASSIFICATION

We now have a training set of 61578 feature vectors and a test set of 79975 feature vectors. Each feature vector is 12000-dimensional, but roughly half of every feature vector is 0 because of the soft-assignment coding scheme.

Coates et al. [4] suggest using a simple linear L2 support vector machine for classification. However, since the data set is rather large, a stochastic gradient descent (SGD) algorithm was used. SGD runs linearly in the amount of samples and is very fast [2]. Note that linear support vector machines also run linearly in the amount of samples, but SGD was chosen because it proved to be slightly faster than support vector machines without losing any accuracy.

LeCun et al. [6] give some practical information on using stochastic gradient descent as a classifier or regressor. They mention a series of data recommendations:

1. The training set should be shuffled;
2. Each feature should have zero-mean;
3. Each feature should have equal variance.

Item 1 is taken care of by the implementation of the stochastic gradient descent algorithm provided by scikit-learn (see Section 5.1). The other prerequisites are met by subtracting the mean from each feature and dividing by its standard deviation.

3.6 WHITENING

Whitening is a preprocessing step to reduce correlation between input values [5]. In images specifically, the adjacent pixel values are correlated. If we whiten the data first, the classification algorithm can be trained on uncorrelated data, which could have beneficial effects on its results.

3.6.1 *Procedure*

Whitening only works if the data matrix X, which contains rows of patches, has a zero-mean. Therefore, our first step is to subtract the mean from each data vector $\mathbf{x}_i \in X$. The length of this vector is $w \times w \times d$, where w is the width (and height) of the patch, and d is the amount of channels. Since we are using RGB images, we have three channels.

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i - \frac{1}{\#\text{pixels}} \sum_{j=1}^{\#\text{pixels}} x_{ij} \tag{3}$$

Our goal is to find the eigenvectors and eigenvalues of the covariance matrix of X. We can use the eigenvectors and eigenvalues to decorrelate our input vectors $\mathbf{x}_i \in X$.

We can easily compute the covariance matrix Σ sequentially on a large data set in two phases. In the first phase we perform the following sums:

$$S = \sum_{i=1}^{\#patches} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T \tag{4}$$

$$\mu = \frac{1}{\#patches} \sum_{i=1}^{\#patches} \tilde{\mathbf{x}}_i \tag{5}$$

In the second phase we can get the final covariance by dividing S by the amount of samples and by subtracting the product of the means of the two data vectors:

$$\Sigma = \frac{1}{\#patches} S - \mu\mu^T \tag{6}$$

Using singular value decomposition we can now find the eigenvectors and eigenvalues of Σ:

$$\Sigma = USV^T \tag{7}$$

We can now compute a matrix P that we can multiply with our original data matrix X to obtain a whitened data matrix X_{white}:

$$P = U \frac{1}{\sqrt{S + \epsilon \times I}} U^T \tag{8}$$

ϵ is a regularization parameter to prevent extremely low eigenvalues from blowing up P.

$$\mathbf{b}_i = \tilde{\mathbf{x}}_i - \mu \tag{9}$$

$$X_{\text{white}} = BP \tag{10}$$

The whitened versions of the centroids as shown in Figure 6 can be found in Figure 7.

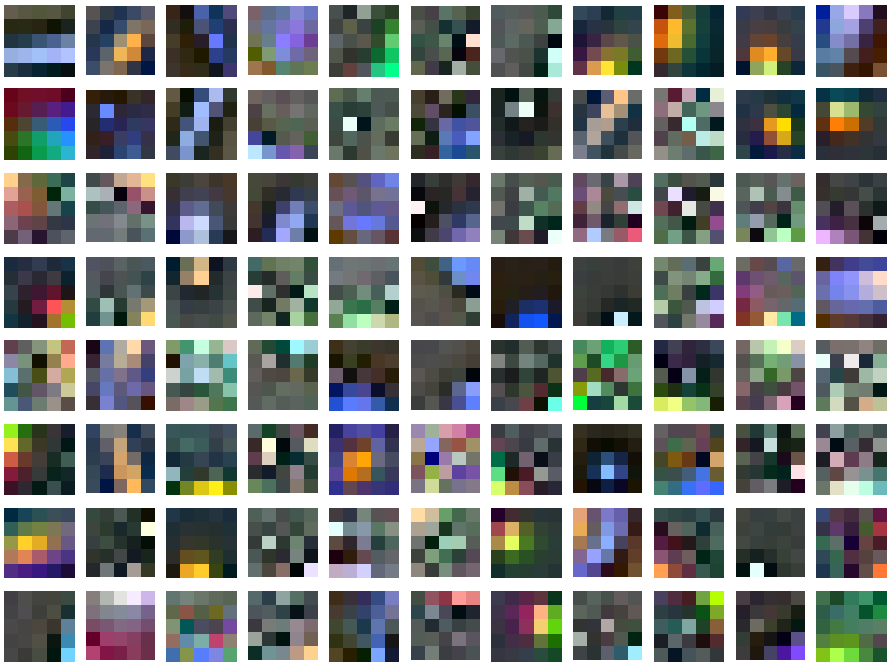


Figure 7: Whitened versions of the random centroids as shown in Figure 6.

3.7 USING THE HIERARCHY

Since there is a hierarchy woven into the classification problem, a regressand can contain information useful for other regressands. However, in our case, all regressands are trained independently, which means the hierarchy is not used at all.

To combat this, a random forest regressor is applied after the SGD regression. It is trained on the predictions of the training images, as generated by the SGD regressor. Now, predictions made by the SGD regressor in the test set can be fed into the random forest regressor. This method is used because using a random forest regressor initially is computationally hard. A random forest regressor is however much faster at predicting, so for this step, where the input is small (only 37 dimensional) it is a better choice than using another SGD regressor. The predictions the random forest regressor makes – based on the SGD regressor test predictions – are the final result that can be uploaded to Kaggle for validation.

4 ROTATION INVARIANCE

4.1 THE PROBLEM

There are three transformations that are relevant for object recognition: rotation, scaling and translation. In the Galaxy Zoo dataset, features are ideally rotation, scale and translation invariant, meaning that if a test image has a different rotation, scale or position than a previously seen training image, the object recognizer will still classify the image correctly. Since patches are extracted randomly during training, the system is already translation invariant. The rotation of galaxies as seen from Earth has no effect on their morphology. Thus if the system is rotation invariant, it should obtain a better score for galaxies in the test set that are very similar to galaxies in the training set but merely have a different rotation.

4.2 THE NAIVE WAY

A trivial way of solving this is to duplicate the training data multiple times, each time rotating it a certain amount of degrees. This effectively increases the training set and consequently reduces overfitting. However, this way of creating a rotation invariant system comes with a major drawback: its time complexity. The running time of the feature learning phase, the feature extraction phase and the classification phase grows polynomially with the amount of rotations (ρ). This is due to the fact that we have increased the size of the data set (ρ-fold), but also because of the need of additional centroids (ρ times as many) to accomodate for the extra features that are now present in the data set.

4.3 MODIFYING THE DISTANCE METRIC

A better way would be to change the distance metric in the k-means algorithm to accommodate for patch rotation. This would remove the need for additional centroids and thus increase the time complexity only linearly in the amount of rotations, as opposed to polynomially if using the naive way of duplicating data. There are two things that need to be changed in order to make the system rotation invariant using this method. Firstly, in the feature extraction phase, the activation has to be calculated differently, specifically not taking into account the rotation of a patch. Secondly, during the training phase, patches have to be assigned to their closest centroid, regardless of the patch

rotation. Both of these changes are fundamentally the same: modifying the distance metric used in the k-means algorithm to be rotation invariant.

4.3.1 *Procedure*

All modifications were made to the k-means implementation in scikit-learn's MiniBatchKMeans[1] class. The method is comprised of the following steps:

1. Rotate every patch in the batch ρ times
2. Compute the Euclidean distance of all rotations of every patch to all of the centroids
3. For each patch, select the rotation that resulted in the smallest distance to a centroid
4. Assign the (possibly rotated) patch to the centroid to which the distance was smallest
5. Update the centroids using only the best rotation of each patch and the centroid it was assigned to

More formally, for each patch:

STEP 1

$$\mathbf{r}_i = \mathrm{rot}(\mathbf{x}, \frac{360i}{\rho}) \tag{11}$$

where $\mathbf{x}$ is the flattened patch vector, ρ is the amount of rotations that are taken into account, rot is a function that rotates a flattened patch a certain amount of degrees and $0 < i < \rho$ where $i \in \mathbb{N}$.

STEP 2

$$d_{ik} = \|\mathbf{r}_i - \mathbf{c}_k\|_2 \tag{12}$$

where d_{ik} denotes the Euclidean distance between rotation $\mathbf{r}_i$ and centroid $\mathbf{c}_k$.

STEP 3

$$\{b, a\} = \arg\min_{\{i,k\}} d_{ik} \tag{13}$$

where a is the index of the centroid the patch is assigned to and b is the index of the best rotation.

1 http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html

STEP 4 AND 5

Update centroid $\mathbf{c}_a$ using patch $\mathbf{r}_b$.

4.3.2 *Feature extraction*

In the feature extraction phase, the activations of all ρ rotations are computed as described in Section 3.4. The rotation that provides the highest activation is selected. Note that the image is rotated in its entirety, not the patches separately. This is done to ensure that the rotation of subimages remains the same throughout the computation of the activation.

4.4 COMPLEXITY

In Table 2 the complexities per phase are shown for the regular method, the naive method and the adapted distance metric method. It shows that while the naive method grows quadratically in the amount of rotations, the modified distance metric method grows only linearly in the amount of rotations. This is due to the fact that we do not need extra centroids in the modified distance metric method.

In the following explanation, i denotes the amount of images, s denotes the size of the image and ρ denotes the amount of rotations that are taken into account. In the modified distance metric method, the amount of patches is the same as in the regular (not rotation invariant) method: is. In the naive method, we need to process $is\rho$ patches, since we duplicate the data and thus have ρ times more patches.

c denotes the amount of centroids that we use in the regular (not rotation invariant) method. In the naive method, we need ρ more centroids, while in the modified distance metric, we need c centroids.

Computing the distance between a patch and a centroid has a constant cost in the naive approach, but has a cost of ρ in the modified distance metric, since we need to compute the distance for every rotation. In the naive method we also need to compute the distance for every rotation, but since the patches were duplicated, we do not need to modify the cost because we can abstract from the fact that the patches are rotations of each other.

For the regular method, the time complexity of the feature learning phase and the feature extraction phase grows with the amount of images, the size of the images and the amount of centroids, thus the complexity is $\mathcal{O}(isc)$. For the modified distance metric, we need to multiply this by ρ because we need to compute distances for every rotation. For the naive method, this is multiplied by ρ^2 because we have ρ more centroids, and we have ρ more patches.

For the training of the classifier, the complexities for the regular method and the modified distance metric are equal. However, since the amount of centroids was multiplied by ρ and the amount of input

vectors was also multiplied by ρ, the complexity of the naive method grows to $\mathcal{O}(ic\rho^2)$.

We assume that the classifier runs linearly in the amount of samples and the amount of iterations for k-means and the classifier is constant for all methods.

	Feature learning	Feature extraction	Classifier training
Regular	$\mathcal{O}(isc)$	$\mathcal{O}(isc)$	$\mathcal{O}(ic)$
Naive	$\mathcal{O}(isc\rho^2)$	$\mathcal{O}(isc\rho^2)$	$\mathcal{O}(ic\rho^2)$
Modified distance metric	$\mathcal{O}(isc\rho)$	$\mathcal{O}(isc\rho)$	$\mathcal{O}(ic)$

Table 2: The rough complexity of each of the phases for several methods. i denotes the amount of images in the initial data set, s denotes the size of each image, c denotes the amount of initial centroids (the amount one would use if there was no rotation invariant algorithm being used), ρ denotes the amount of rotations that are taken into account.

4.5 REMARKS

Note that it is only possible to find four rotations of a square image without having to extrapolate pixels. In the following chapters, ρ is assumed to be equal to 4. In Section 7.5 a method by Charalampidis [3] is explained that can efficiently work with more than four rotations.

5 RESEARCH AND EXPERIMENT

5.1 IMPLEMENTATION

Most of the implementation was done using the scikit-learn Python library and MATLAB. MATLAB was chosen for the feature extraction phase, since using the GPU is very easy in MATLAB. Using the GPU decreased the running time significantly to about 20% of the original running time. The feature training and classification phases were not very suitable for GPU processing, since memory on the device is limited and both phases require a lot of memory compared to the feature extraction phase.

Especially useful were the following scikit-learn modules:

- sklearn.cluster.MiniBatchKMeans
- sklearn.linear_model.SGDRegressor
- sklearn.ensemble.RandomForestRegressor

The repository containing all code used in this thesis can be found on GitHub: `https://github.com/StevenReitsma/galaxyzoo/`.

5.2 WORKING WITH BIG DATA

Since the data set is too large to fit into memory, some of the techniques used by Coates et al. [4] had to be adapted to allow for sequential or batch processing. Examples of this are the MiniBatchKMeans module, which can run k-means on separate batches of input data, but also the computation of the covariance matrix for the whitening step as described in Section 3.6. With these adjustments, the entire pipeline can be run on a computer with just 4 gigabytes of memory. This prevents the need of a cluster or a computer with a higher (32GB+) amount of memory, which can be costly.

5.3 INTERESTS AND EXPERIMENT SETUP

The first experiment is to test whether the rotation invariant distance measure as described in Chapter 4 improves the final result. Furthermore, the effects of whitening, the patch size and the amount of centroids are tested. Because the duration of each test is longer than 24 hours, the amount of tests has been kept to a minimum.

The reported scores are all obtained by uploading the result to Kaggle. However, this does not provide us with the variance of the scores

which makes it harder to do a statistical test to determine significance. The variance is therefore estimated using a small portion of the training set, of which the results are known and of which the mean and variance can be calculated. This portion of the training set was not used for training.

In the next section, the results are shown.

6 RESULTS

6.1 COMPARED TO BENCHMARKS

In Figure 8 the root mean squared errors of the benchmarks and contest winner are shown, together with the score obtained using the method as described in this thesis.

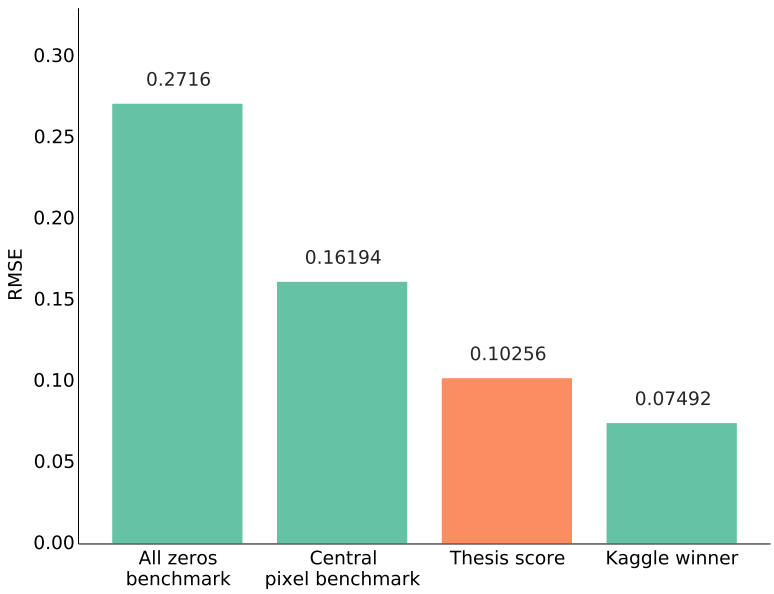


Figure 8: Root mean squared error for several benchmarks, the thesis score and the score obtained by the winner of the Kaggle competition.

6.2 ROTATION INVARIANCE

In Table 3 the root mean squared errors of the two pipelines are shown. The difference appears to be small, but it is significant with a p-value smaller than .001. As mentioned in Section 5.3 the variance is estimated to determine this p-value. The estimated variance values are shown in Table 4. However, since the sample size is so large (79975 galaxies), even if these estimations are off by over a factor of 10, they would still result in a p-value below the threshold of .05. In Figure 10 the difference in root mean squared error is shown for each separate question (questions are shown in Section 2.2.1).

Regular	0.10789
Rotation invariant	0.10256

Table 3: Root mean squared error for the regular pipeline and the modified pipeline as described in Chapter 4. $\rho = 4$

Regular	0.055823
Rotation invariant	0.048997

Table 4: Standard deviation of the error for a subset of the training data (held back from the training procedure itself). These values are used to determine the standard deviation in the test set to estimate the p-value.

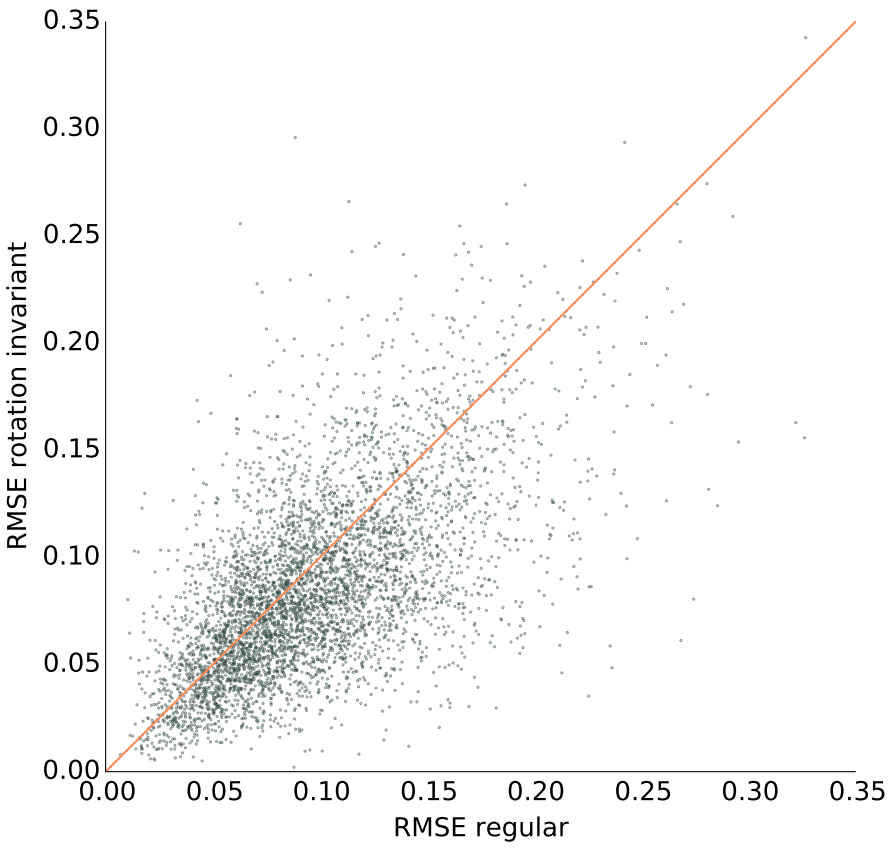


Figure 9: Scatter plot of the root mean squared error of 5000 samples of the regular pipeline and of the rotation invariant pipeline. There are a total of 3179 data points below the $x = y$ line. This shows a significant improvement with a p-value $< .001$.

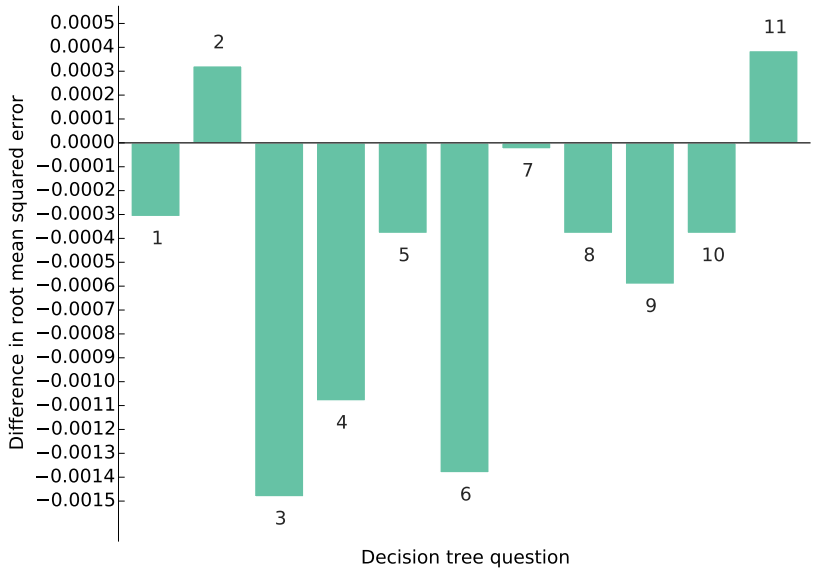


Figure 10: The difference between the root mean squared error of the regular method and the rotation invariant method for each question in the decision tree as shown in Section 2.2.1. Negative values mean that the rotation invariant method performed better for that question. Note that the predictions were scaled to sum up to 1 for each question in this graph in order to make a fair comparison.

6.3 WHITENING

In Figure 11 the results for the whitening step are shown.

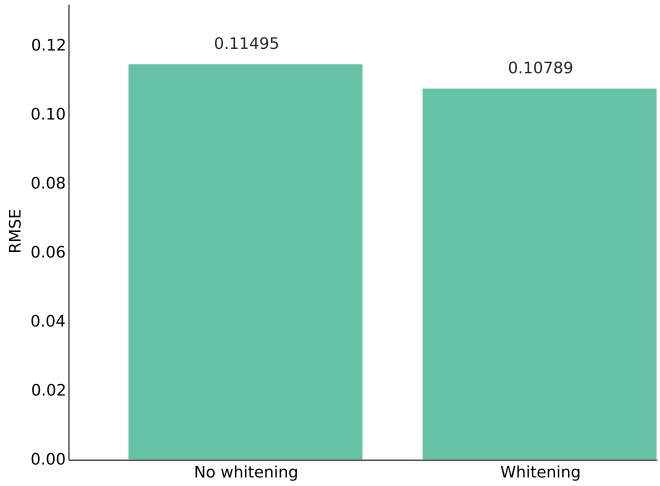


Figure 11: Effect of whitening on the root mean squared error. The reported results were obtained without using the rotation invariant distance metric.

6.4 PATCH SIZE

In Figure 12 the results for various patch sizes are shown. Not all patch sizes between 3x3 and 10x10 were tested due to high computation times.

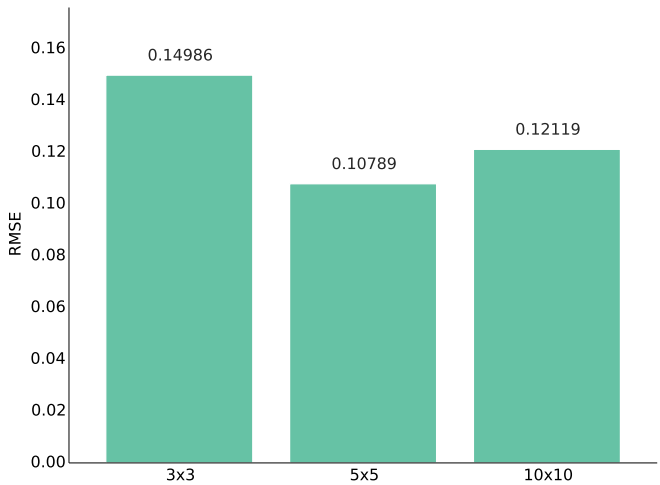


Figure 12: Effect of the patch size on the root mean squared error. The reported results were obtained without using the rotation invariant distance metric. Whitening was used.

6.5 AMOUNT OF CLUSTERS

In Figure 13 the results for different amounts of clusters are shown. Again, not all cluster amounts were tested due to high computation times.

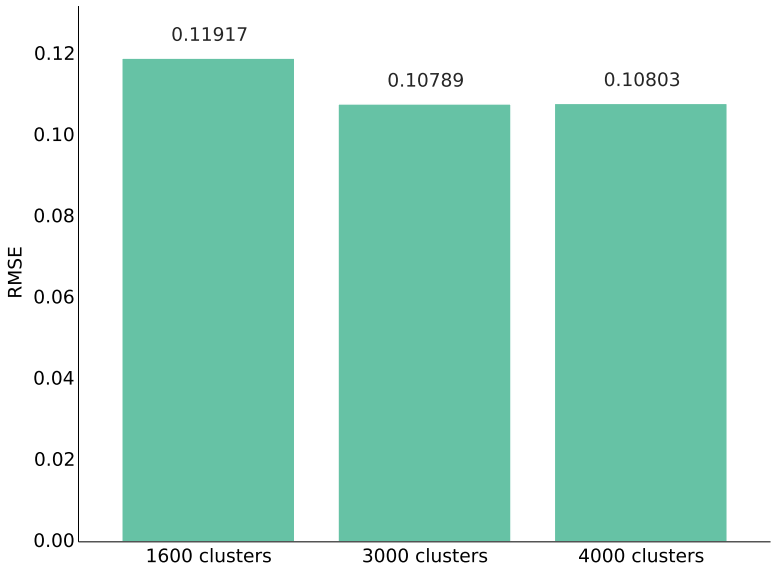


Figure 13: Effect of the amount of clusters on the root mean squared error. The reported results were obtained without using the rotation invariant distance metric. Whitening was used. A patch size of 5x5 was used.

7 CONCLUSION AND EVALUATION

7.1 WHITENING

As shown in Figure 11, whitening significantly reduces the error. This matches the conclusion reached by Coates et al. [4]: that whitening is a crucial step since k-means is blind to correlations in data.

7.2 PATCH SIZE

Of the tested patch sizes, a 5x5 pixel patch size performs best, as shown in Figure 12. It seems that patches of 3x3 pixels are too small to hold features and patches of 10x10 pixels are too big and as a result might hold multiple features. Using more centroids could improve the results of suboptimal patch sizes.

7.3 AMOUNT OF CLUSTERS

As shown in Figure 13, increasing the amount of clusters from 1600 to 3000 shows a significant improvement. However, increasing the amount of clusters even more, to 4000, does not show a significant difference in the error.

7.4 ROTATION INVARIANCE

Since the p-value indicates a significant difference between the root mean squared error of the regular pipeline and the error of the rotation invariant pipeline, and the difference is in the right direction, we can conclude that the rotation invariant pipeline is better at classifying the morphology of galaxies. Generalizing features (e.g. by making them rotation invariant) to detect variants of a patch seems to be good practice in certain domains, certainly for galaxy morphologies.

In Figure 10 it is shown that the rotation invariant method is better for most questions but is worse for others. The two questions for which the rotation invariant method is worse are the following:

- Can the galaxy be viewed edge-on?
- How many spiral arms are there?

One possible explanation for this is that when dealing with spiral arms, the rotation is important. Consider the example where there are two spiral arms. When rotating one of the spiral arms, it will

form a perfect overlay on the other spiral arm. For the rotation invariant algorithm, these spiral arms are thus encoded by the same centroid. This will result in a higher activation in the feature extraction phase, and due to the soft activation function should result in a higher value in the final feature vector. However, during the feature extraction phase, the activations of an image are calculated without rotating subregions of the image. The image is always rotated as a whole. Normally, this is desired – since otherwise an image with a rotated subregion would have the same activation as one with an unrotated subregion – but for the spiral arm question, this way of computing the activations does not produce good results. This is due to the fact that the features (centroids) are still rotation sensitive. They encode for only one of the rotations, it is just the distance metric that makes the system rotation invariant. If an image contains two spiral arms, only one of them can be detected, since the image and its 180 degree rotation will both have the same activation for the spiral arm. It cannot be detected twice because the image is rotated as a whole. In the regular method this does not form a problem since the different spiral orientations in an image are encoded by different centroids altogether.

A possible solution would be to use the regular results for the questions above, and the rotation invariant method for the others. This could improve the performance even more.

7.5 FURTHER WORK

As explained in Chapter 3 the size of the images was reduced from 424x424 to 15x15 pixels because of the high running times of the system. This reduction will definitely have impacted the performance of the system and it would be interesting to see how the system would perform with a decreased reduction factor.

7.5.1 *Circular shift invariant k-means*

Charalampidis [3] modified k-means so that it is circular shift invariant. A circular shift can be seen as a function that shifts all values in a vector n places, for example, where $\mathbf{x} = (1, 2, 3, 4, 5)$:

$$\mathrm{circ}_1(\mathbf{x}) = (2, 3, 4, 5, 1) \qquad \mathrm{circ}_2(\mathbf{x}) = (3, 4, 5, 1, 2) \tag{14}$$

In Cartesian coordinates, a circular shift does not correspond to a rotation and will be useless to make the system rotation invariant. However, if we convert the Cartesian coordinates to polar coordinates first, circular shifts *will* correspond to rotation. The advantage of using the system Charalampidis [3] developed is that it is much more efficient and thus can take into account more rotations than the four

used in the research of this thesis. It would be interesting to see the effects of using more rotations.

7.6 FINAL REMARKS

In Chapter 3 and Chapter 4 some theory background is given on unsupervised learning using k-means and adapting the distance metric to account for rotation sensitivity within the system. In Chapter 6 it is shown that making the system rotation invariant improves its performance significantly.

There still exists a gap between the winner of the Kaggle competition and the score achieved using unsupervised learning as described above. However, this system has very few hyperparameters and runs much faster than the deep learning techniques utilized by the winner and runner-ups.

Using the Kaggle challenge, the Galaxy Zoo team has shown that the performance of automatic classification is improving. Whether automatic classification is going to replace manual classification or crowd sourcing in its entirety any time soon remains to be seen.

BIBLIOGRAPHY

[1] Kevork N Abazajian, Jennifer K Adelman-McCarthy, Marcel A Agüeros, Sahar S Allam, Carlos Allende Prieto, Deokkeun An, Kurt SJ Anderson, Scott F Anderson, James Annis, Neta A Bahcall, et al. The seventh data release of the Sloan Digital Sky Survey. *The Astrophysical Journal Supplement Series*, 182(2):543, 2009.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186, 2010.

[3] Dimitrios Charalampidis. A modified k-means algorithm for circular invariant clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(12):1856–1865, 2005.

[4] A Coates, AY Ng, and H Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[6] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. 2012.

[7] Iskra Strateva, Željko Ivezić, Gillian R Knapp, Vijay K Narayanan, Michael A Strauss, James E Gunn, Robert H Lupton, David Schlegel, Neta A Bahcall, Jon Brinkmann, et al. Color separation of galaxy types in the Sloan Digital Sky Survey imaging data. *The Astronomical Journal*, 122(4):1861, 2001.

[8] JC Van Gemert, J-M Geusebroek, CJ Veenman, and AWM Smeulders. Kernel codebooks for scene categorization. *Computer Vision–ECCV 2008*, pages 696–709, 2008.

[9] Kyle W Willett, Chris J Lintott, Steven P Bamford, Karen L Masters, Brooke D Simmons, Kevin RV Casteels, Edward M Edmondson, Lucy F Fortson, Sugata Kaviraj, William C Keel, et al. Galaxy Zoo 2: detailed morphological classifications for 304122 galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 2013.

[10] Donald G York, J Adelman, John E Anderson Jr, Scott F Anderson, James Annis, Neta A Bahcall, JA Bakken, Robert Barkhouser,

Steven Bastian, Eileen Berman, et al. The Sloan Digital Sky Survey: Technical Summary. *The Astronomical Journal*, 120(3):1579, 2000.