BSc Honours Computer Science Research Project
Galaxy Zoo: Data Augmentation and Loss for Galaxy Morphology Classification

Chloë Smith and Ritesh Ajoodha

School of Computer Science and Applied Mathematics. University of the Witwatersrand, Johannesburg. South Africa
1877342@students.wits.ac.za,Ritesh.Ajoodha@wits.ac.za

Abstract. With the increased volume of galaxy image data, there is a great need for automatic annotation, necessitating a solution for image classification of galaxy images by their morphologies. Galaxy morphology classification has traditionally been done by hand, with the largest annotated datasets having been produced by the Galaxy Zoo citizen science project. In this paper, we look at using the annotations produced by citizen science to fuel automation, investigating the effects of data augmentation/transformation on the performance of a deep learning model on this problem. We use a modified VGG-16 architecture and attempt to find a relatively lightweight solution using different loss functions, and data augmentation in the form of image transformation. We find that data augmentation achieves immediate performance improvements, and that robust loss functions

Keywords: computer vision, galaxy morphology classification, image classification, deep learning, transfer learning, data augmentation, loss functions

1 Introduction

Classification of galaxies by their morphologies has been an important problem for as long as we have been able to obtain images of them. Historically, this has been done by the hand of experts, and more recently via citizen-science projects such as Galaxy Zoo, where a group of expert and amateur volunteers answer questions about these images by eye. With the increase in volume of these images, this sort of manual classification has become infeasible, and the problem of automatic classification ever more relevant.

While the problem of image morphological classification has largely been solved to a high accuracy with quite efficient models, there are many possible permutations of even just neural-network (NN) related solutions, and in this paper, we explore the effects of feature selection and data augmentation on the problem of galaxy morphology classification, in the context of the Galaxy Zoo project and competition.

1.1 Related Work

The most relevant datasets to this topic are in the set of various citizen-science projects carried out by Galaxy Zoo. Some of the research under consideration uses the images collected in the Sloan Digital Sky Survey (SDSS) prior to 2016 [1] and some use more recent images from the Dark Energy Camera Legacy Survey (DECaLS) [2].

The prior research uses this data in various ways and to varying degrees. For instance, the first published Galaxy Zoo dataset (henceforth referred to as GZ1 or GZ1-T2), has been used considering just two classes, elliptical or spiral [3], further dividing the data into expert-labelled (GZ1-E) and amateur-labelled (GZ1-A) sets, for a total of 41 424 examples. The winning entry of the Kaggle Galaxy Zoo competition uses the GZ1-T2 dataset used in the competition [2][4], removing over-represented morphological categories to result in a dataset of 61 578 JPEG images, used this way in other works as well [5]. The EFIGI catalog, containing images from the SDSS [6], and the GZ1 dataset augmented with redshifts from SDSS have been classified as well [7], as well as the more recent GZ-2 dataset [8].

There is also a subset of the research that does not use the volunteer-labelled images of the Galaxy Zoo projects at all, and one considering a different field: classifying the charge-coupled device (CCD) images from the Digitised Second Palomar Observatory Sky Survey (DPOSS) dataset [9], as well as CCD images [11] of the morphological types of 1857 astronomical objects [10].

The most commonly used models appear to be deep learning (DL) based models. This method is useful as it does not necessarily require the manual extraction/selection of features (FS) from the image, but can also be used in the process of doing so.

Considering experiments on the Galaxy Zoo dataset: auto-encoders (AEs) and a feature-extraction method called WND-CHARM [13] have been compared [3]. Features are ranked based on Fisher scores, and the lowest ranking are discarded. Another strategy is to exploit the rotational invariance of galaxy morphologies, producing 'viewpoints' from each image by a process of cropping, scaling, rotation, brightness adjustment and re-centering [14], resulting in 45x45 pixel RGB images, which were the inputs for the CNN model used [4]. Similarly, a method has been described in which the images are cropped, downscaled, rotated, flipped and colour-adjusted to produce 64x64 pixel RGB images for input to the CNN model [5].

Multiple works use convolutional layers for implicit feature extraction [4][5], which can then be fed into dense layers. One convolutional layer with filters can be used for feature extraction [6], whose output is then fed into 24 fully-connected layers for classification. Photometric and spectra values have also been used as features [7], which are based on the visual characteristics of the galaxy images (colour, shape, etc.), with the GZ1 data used to test the ML models considered. This research took a decision-tree based approach, using the random forest (RF) classification model, varying the number of trees and spectra band features used, with RF models being a somewhat popular approach [8], possibly

for their interpretability and closeness to the original method of data gathering. Some of the algorithms used here include GID3* [9][16], a generalisation of the popular ID3 algorithm, as well as the O-Btree algorithm [17], which deals with feature selection on the basis of impurity features, which aim to partition the set into desired classes and largely ignore any differences within the classes.

Some works take a more qualitative approach, using mathematical morphology (MM) techniques to classify the images [10]. Rather than choosing features, MM uses 'structuring elements' to analyse the image, using operators that produce quantities that can then be compared between examples.

The basic implementation of an algorithm can sometimes be inadequate or have unexpected results for the specific problem/data at hand. In this section, we look at some of the ways prior research has dealt with optimising their respective models to increase accuracy and/or reduce run-time.

A comparative study was performed to determine the best performing model [3], considering the following classic ML algorithms:

- **k-nearest neighbours (kNN)**, using Euclidean distance and k=3,
- **Random forests (RF)** with a number of sub-trees n = 100, using the Gini criterion, and
- **Support vector machine (SVM)** using a linear kernel type, a tolerance of 10^{-4} and C = 1.0.

Two autoencoder (AE) models for feature selection were implemented and compared:

- a **Deep AE (DAE)**, with two fully-connected layers, the output having lower dimension than the input (256 features), and
- a **Convolutional AE (CAE)** which uses convolutional and pooling layers to produce an encoding of the input images, using 3 pairs of convolutional and pooling layers with ReLU activation units.

The DAE described as well as deeper architectures were tested, and it was observed that increase in accuracy was insignificant and for some architectures, even less than the CAE, so the CAE was chosen for further comparison.

A CNN architecture was proposed, consisting of three pairs of convolutional and pooling layers, feeding into two dense layers to produce the classification at the output layer, using ReLU activation units in all layers but the final dense layer, which used softmax activation units.

Combinations of the FE methods, (DAE, CAE and WND-CHARM in greyscale and colour), FS methods (none, Fisher scores, embedded FS), and the ML models (kNN, RF, SVM and the proposed CNN) were implemented, and it was found that AEs performed the FE task in the least amount of time, while WND-CHARM provided better accuracy for the GZ1 dataset. Of the classic ML algorithms, RF performed the best overall, and the use of colours also increased accuracy, at a cost of higher run-time. The FS methods did not significantly increase accuracy, but they did reduce prediction run-time. CNNs performed the

best overall, and it was shown that increasing depth and complexity of a CNN model does not always yield proportionate improvements in performance.

Overfitting is addressed through the use of data augmentation (producing 16 viewpoints of each image)[4], regularisation, parameter sharing (via exploiting rotational and translational symmetry in the input), and model averaging (take advantage of the strengths of each model, and reduce variance), as well as dropout. A CNN consisting of 7 hidden layers, 4 convolutional with ReLU activations and 3 dense with maxout activations, were trained using mini-batch gradient descent.

Scale jittering is used for data augmentation (images are scaled and cropped to produce new images)[5], and use a variant of the ResNet architecture consisting of 26 convolutional and pooling layers, using softmax activations, and trained with mini-batch gradient descent with a variable learning rate, but some works emphasise the choice of features for optimisation [7].

Deep learning and CNN approaches specifically have shown promising results in this domain [4][5][6][12], with some achieving better results using a decision-tree based model [9], but this is very likely due to the computational power available at the time, and it was found that while MM yielded excellent results on CCD images, this did not generalise well to photographic images [10]. The best results appear to come from simplifying the problem to differentiating just two classes, approaching perfect accuracy using deep learning, and 82% when differentiating three classes [18].

As we can see, the body of research on galaxy morphology classification and image classification in general is vast, but there is still much room for experimentation. It is almost impossible to test each and every permutation of model and parameters possible, but any valid experiment can provide direction to practitioners and researchers in this domain. We specifically aim to show the effects of different loss functions on a model for galaxy morphology, as well as in-place data augmentation.

2 Method and Model

In this section, we describe the techniques used to arrive at our results, including transfer learning, augmentation/transformation of the data, and the training/optimisation methods and the hyperparameters tested. We specifically focus on the effect of different loss functions on the accuracy of our model, with or without transformations to the images. The code used for this project can be found at: `https://github.com/cds21199/research2021/blob/main/GalaxyMorphologyClassification.ipynb`.

2.1 Dataset

The data used for this project was sourced from the Kaggle Galaxy Zoo competition [1], we use the RGB images in the training set (61578 images) with a validation split of 20%. In this experiment, we use no explicit feature extraction,

opting to let the CNN perform implicit FE. The Kaggle dataset includes a CSV file containing the values of our 37 targets for each image.

2.2 Data augmentation/Preprocessing

Data augmentation is traditionally described as an additive process - transforming data and appending these results to the original data for training. In our case, we are using augmentation in an in-place form, never training on the original images but on their transformed versions. With a dataset of nearly 70 000 images, this may be sufficient.

Considering that galaxy morphologies are translationally and rotationally invariant, we use the following transformations to produce the input to our model, after resizing to 224-by-224 pixel images:

- Random horizontal shifts within a range of 0.2 in either direction;
- random vertical shifts within a range of 0.2 in either direction;
- zooming (scaling and cropping) within a range of 0.2, in or out;
- some images are randomly flipped horizontally and/or vertically.

For transformations that require extrapolation of pixel data at the edges, we use nearest extrapolation, where a pixel is filled in with the value of the pixel closest to it.

2.3 Models: Benchmark and Transfer Learning

For benchmarks, we looked at two models: one using transfer learning, and the other trained from scratch. For the transfer learning model, ResNet weights trained on ImageNet were used, freezing the pre-trained layers, and adding a global pooling layer, three fully-connected layers for training and a final fully-connected layer with 37 softmax units for our regression, resulting in 2,763,813 trainable parameters out of 26,351,525 parameters. For the custom model, we use a CNN architecture similar to VGG-16, with a total of 30,361,573 trainable parameters, consisting of the following:

- Two groups of two 2D convolutional layers using ReLU activations followed by a max-pooling layer with dropout,
- Two groups of three 2D convolutional layers using ReLU activations followed by a max-pooling layer with dropout,
- The output is then flattened and fed into three fully-connected layers using using ReLU activations and dropout
- The label predictions are produced by a final fully-connected layer with softmax activation.

The standard hyperparameters for each model are described in Table 1

The ResNet model with no augmentation and mean square error (MSE) loss took a very long time to run per epoch, and also seemed to plateau very early with a low accuracy (just under 60%). We move forward with the custom model, which produced training accuracy of 68.10% and validation accuracy of 72.16%.

Table 1. Hyperparameters

Parameter	**Details**
Optimizer	Gradient Descent
Initial Learning Rate	0.1
Momentum	0.9
Learning Rate Schedule	exponential decrease after 10 epochs
Dropout*	0.25

*where used

2.4 Loss functions

Using the CNN architecture described above, we investigate the effects of a few loss functions for regression, and their implications on training a model with outliers - in the context of our dataset, we have a mixture of amateur and expert labels, meaning that we can expect some disagreement. A brief description of the loss functions used follows below:

- **Mean Squared Error (MSE)**:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{1}$$

  MSE, as a square loss function, heavily penalises large errors, and is measured in the square units of our labels.
- **Root Mean Squared Error (RMSE)**:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{2}$$

  This is very similar to Euclidean distance between two vectors, estimating the standard deviation of our model's predictions versus the true labels. This is measured in the same units as our labels, as opposed to MSE.
- **Huber loss**:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & otherwise \end{cases} \tag{3}$$

  Squared loss can be heavily influenced by outliers; Huber loss seeks to tackle this using a threshold, above which the error is no longer squared, but multiplied by a fixed threshold. This has the effect of flattening the function or reducing its steepness.
- **Mean Squared Logarithmic Error (MSLE)**:

$$\frac{1}{2}\sum_{i=1}^{n}(log(x_1 + 1) - log(y_i + 1))^2 \tag{4}$$

  Since here we are using the logarithms of the errors, we are scaling them down and should therefore increase robustness of our loss function to outliers.

2.5 Limitations

We are limited in this experiment by both scope, and time. Considering the depth of the models used and training time, we chose a lightweight architecture that would allow for many executions to cover each of the variables considered, namely data augmentation and the different loss functions. In terms of computing power, we aimed to design a model that could feasibly be run using Google Colaboratory, which proved to be difficult and the experiment necessitated the use of Colab Pro, for the increased disk allowance and GPU runtime. We use the entirety of the 37 targets available, this is likely to be a source of the somewhat mediocre accuracy results, as we have a very high-dimensional target but a relatively shallow architecture, and using in-place data augmentation meant that we could not use a much more data hungry network.

3 Results and Analysis

The results of each model are presented in Tables 2 and 3. The first change we make is using the augmented input, and see that the best model using this method gives an immediate improvement of more than 7% in training and 4% in validation accuracy, and the improvement came much sooner in training. We look to improve upon this, using the different loss functions described above, and see that RMSE gave us the best performing model, with an improvement in validation accuracy of more than 2% on MSE, and roughly 4% on the benchmark model with no augmentation.

Each model was run for a minimum of 30 and a maximum of 50 epochs if they did not converge, and run-time was on average 18 hours. Possibly as a result of the learning rate schedule used, running the models for more than 50 epochs yielded very little, if any improvement. In fact, past the 30 epoch mark, a model would not improve by more than 1%, and the loss function would stagnate between 0.2 and 0.4. We note that the most conservative or robust of our loss functions learned the slowest, which is not unexpected.

Looking at the results of the ResNet transfer learning model, this took 10 hours to train and converged to a mediocre result very early on. Considering that the pre-trained weights used were on ImageNet, i.e., images of ordinary objects, it is not surprising that this did not easily extend to astral objects. In this case, it is likely that to make this particular model work, we would need to add many more layers, but this is just compounding a problem that can easily be solved by training a new model. We achieved much better results from a less deep network, showing that we can't take for granted that pre-trained weights on a particular task will transfer well to a new domain.

From the perspective of looking at pure effects of the loss functions and data augmentation, it is clear that exposing the model to valid transformations of the images significantly increases accuracy, even when we do not use this method to increase the volume of training data, and with the hyperparameters used, more conservative loss functions converged earlier than the more commonly used

Table 2. Training and Validation Accuracy - Benchmark Models*

Model	Converged? (else, epochs)	Training Acc. (%)	Valid. Acc. (%)
ResNet Transfer Learning	yes	59.69	59.47
VGG-16-like CNN	10	68.10	72.16

*these models were run with MSE loss, no augmentation

Table 3. Training and Validation Accuracy - with Augmentation*

Loss function	Training Acc. (%)	Valid. Acc. (%)
MSE	72.97	73.99
RMSE	75.49	76.15
Huber Loss	70.38	73.71
MSLE	67.11	72.40

*run using gradient descent, lr = 0.1 with exponential scheduling, momentum = 0.9

RMSE. In a large portion of prior work that achieved very high accuracy, the scope was decreased to focus on only a handful of classes, and our goal of fitting a model to all 37 targets is achieved to some degree, but with much room for improvement.

4 Conclusion

For our relatively high-dimensional regression problem (37 targets), RMSE was found to achieve the best results in a relatively short amount of time, with a deep, but not extremely deep, CNN architecture. An area for further experimentation could be to reduce the problem size, say to classifying images into just elliptical or spiral morphologies, and then using pre-trained weights on that problem to extend to the broader 37-question regression problem. In-place data augmentation showed great improvements, in both run-time and accuracy, and we see that the increase in performance outweighs the loss of information when reducing the scale of these images. A high amount of detail is not completely necessary to identify the key attributes of these images.

References

1. D. Harvey et al. Galaxy Zoo - the Galaxy Challenge. Research Prediction Competition. Kaggle, 2014. URL: https://www.kaggle.com/c/galaxy-zoo-thegalaxy-challenge/overview/the-galaxy-zoo-decision-tree.
2. Campbell Allen, Chris Lintott, and Grant Robert MacKinnon Miller. The science behind the site. Galaxy Zoo. URL: https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/about/research

3. Manuel Jimenez et al. “Galaxy Image Classification Based on Citizen Science Data: A Comparative Study”. In: IEEE Access 8 (2020), pp. 47232–47246.
4. Sander Dieleman, KyleW.Willett, and Joni Dambre. “Rotation-invariant convolutional neural networks for galaxy morphology prediction”. In: Monthly Notices of the Royal Astronomical Society 450 (2015), pp. 1441–1459.
5. Xiao-Pan Zhu et al. “Galaxy Morphology classification with deep convolutional neural networks”. In: Astrophysics and Space Science 364 (2019), pp. 1– 15.
6. Nour Eldeen M. Khalifa et al. “Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks”. In: SRGE (2017).
7. Adam Gauci, Kristian Zarb Adami, and John Abela. “Machine Learning for Galaxy Morphology Classification”. In: Monthly Notices of the Royal Astronomical Society 000 (2010), pp. 1–8.
8. Melanie R. Beck et al. “Integrating human and machine intelligence in galaxy morphology classification tasks”. In: Monthly Notices of the Royal Astronomical Society 476.4 (2018), pp. 5516–5534.
9. Nicolas Weir, Usama M. Fayyad, and S. Djorgovski. “Automated Star/Galaxy Classification for Digitized POSS-II”. In: The Astronomical Journal 109.6 (1995), pp. 2401–2414.
10. Jason A. Moore, Kevin A. Pimbblet, and Michael J. Drinkwater. “Mathematical Morphology: Star/Galaxy Differentiation & Galaxy Morphology Classification”. In: Publications of the Astronomical Society of Australia 23.4 (2006), pp. 135–146.
11. Ian Smail et al. “A Catalog of Morphological Types in 10 Distant Rich Clusters of Galaxies”. In: The Astrophysical Journal Supplement Series 110.2 (1997), pp. 213–225. DOI: 10.1086/312997. URL: https://doi.org/10.1086/312997.
12. C.-C. Yang et al. “Application of artificial neural networks in image recognition and classification of crop and weeds”. In: Canadian Biosystems Engineering 42 (2000), pp. 147–152.
13. Nikita Orlov et al. “WND-CHARM: Multi-purpose image classification using compound image transforms”. In: Pattern Recognition Letters 29.11 (2008), pp. 1684–1693.
14. E. Bertin and S. Arnouts. “SExtractor: Software for source extraction”. In: Astronomy and Astrophysics Supplement 117 (1996), pp. 393–404.
15. C. Scarlata et al. “COSMOS Morphological Classification with the Zurich Estimator of Structural Types (ZEST) and the Evolution Since z = 1 of the Luminosity Function of Early, Disk, and Irregular Galaxies”. In: The Astrophysical Journal Supplement Series 172 (1 2007), pp. 406–433.
16. U. M. Fayyad and K. B. Irani. “KB: A Machine Learning Algorithm (GID3*) for automated mated knowledge acquisition improvements and extensions”. In: General Motors Reseat Report CS-634 (1991).
17. U. M. Fayyad and K. B. Irani. “The Attribute Selection Problem in Decision Tree Generation”. In: AAAI-92 Proceedings (1992).
18. P.H. Barchi et al. “Machine and Deep Learning applied to galaxy morphology - A comparative study”. In: Astronomy and Computing 30 (2020), p. 100334. ISSN: 2213-1337. URL: https://www.sciencedirect.com/science/article/pii/S2213133719300757.