ISSN No:-2456-2165

Morphological Taxonomy of Galaxies Using Convolutional Neural Networks

Aditi Ravishankar[1]*, Aishwarya K Karanth[1]*, Ameena[1]*, Adhishreya P[1]*, Apoorva N Hegde[2]*
[1] Student, Department of Electronics and Communication Engineering, RNSIT, Bangalore, India
[2] Asst. Professor, Department of Electronics and Communication Engineering, RNSIT, Bangalore, India

Abstract:- There are millions of huge collections of stars, gas, dust and stellar remnants all held together by gravity in our vast universe. These collections, or galaxies, help in deciphering the structure and history of the universe in general. The classification of these galaxies based on morphological parameters is a relevant requirement in understanding their formation and evolution. Manual identification of the categories to which each belongs to can be tiresome, time consuming and error prone. The objective of our work was to automate the process of finding the features that characterize a galaxy using convolutional neural networks, a cardinal concept in the image data space, whilst comparing the accuracy of the classification with and without prior processing of the image dataset. The Galaxy Zoo dataset was used for the same and it was preprocessed by applying median filtering and contrast limited adaptive histogram equalization. The final classification model was a CNN based on the VGG-16 architecture with some modifications. We considered all 37 features as per the decision tree by Willet et. al. and with a multi-regression approach, obtained a model with a validation loss of 0.0102 (mean square error) on processed images as the best performing model. The model was then deployed onto a client-side interface using Flask to predict the features of the galaxies in real-time.

Keywords:- *Deep Learning, Image Processing, Convolutional Neural Networks.*

I. INTRODUCTION

Galaxies are the fundamental elements of the universe and the understanding of their origin and evolution is an integral part of physical cosmology. There are an uncountable number of galaxies in the cosmos, with billions not even discovered yet. The structural or morphological properties of galaxies hold the key to the deeper understanding of the process of galactic evolution. The study of the same comprises a rich history in the field of astrophysics, while answering many questions about the universe as we know it. The databases are generated by telescopes at an ever-increasing speed which reinforces the need for computer analysis of the images. The expanding image databases of astronomical objects such as galaxies render the process of manual methods for classification of the same as almost impossible. Experts would require enormous amounts of time to manually categorize the images of the galaxies or find the finer physical features that describe a galaxy. This led to a dire need for the creation of automated means to do the same. Some of the most famous datasets used by related works include the Galaxy Zoo dataset, and the Sloan Digital Sky Survey (SDSS) which is possibly the largest astronomical survey. Other data sources include - Zsolt Frei's catalogue [3], EFIGI catalogue [8], the NGC catalogue [16], etc. Some of them went with a custom dataset constituting a random collection of images from Google [1] and other sources. After the collection of the images, the next step is to rid the images of any unwanted and hindering distortions that exist. Image preprocessing is a vital step in image enhancement which aids in the deduction of important features and inferences. Another important step is data augmentation which is performed in order to introduce more variety into the training data for the reduction of overfitting. Once the data is well prepared comes the actual process of classification. This process involves the fields of Machine Learning and Deep Learning, the flag-bearers of automating methods that require human intervention. The related works have used multiple algorithms of ML and DL. The ML methods previously tested include Support Vector Machines, Random Forest classifier, Naive Bayes classifier, K-Nearest Neighbors and Adaboost classifier [2],[3]. The DL algorithms in the related works are Artificial Neural Networks [15], and mainly different architectures of Convolutional Neural Networks such as Inception Module [1], Resnet [9], etc. Performance wise, DL fared better than ML. Some works worked on a three-category classification (Elliptical, Spiral, Irregular) [1] and some others worked on a five-category classification [2], [9]. A few works have used the Willett et.al. decision tree [21] for attempts in discovering the finer features that describe a galaxy, but some of them have cut the tree short so as to not consider all the possible features. This decision tree was also used as a reference in the famous Galaxy Zoo challenge on Kaggle which over 326 teams participating in the classification of galaxies. Our work on galaxy morphology classification is described in the following sections, which include the dataset under consideration, the preprocessing techniques involved, the models trained and its performance. The objective of this work is to achieve a model with a low loss using Convolutional Neural Networks, a cardinal concept in image.

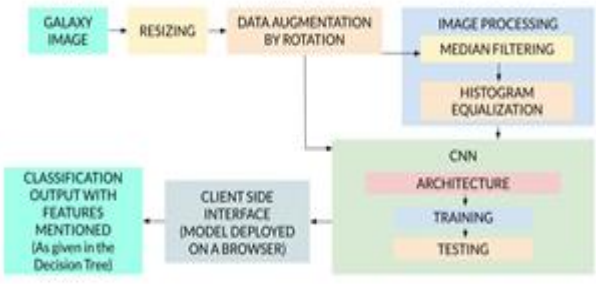

Fig 1: System Block Diagram

Classification, to predict the probabilities with which features characterise a given galaxy, with a multi-regression approach to the problem statement. The system block diagram is shown in Fig 1.

II. DATA

The Sloan Digital Sky Survey has mapped nearly one-third of the sky which lead to the Oxford University possessing a dataset of millions of images of galaxies that needed to be classified based on their morphologies to better understand the galactic processes. Classifying images at this scale would indeed require a significant amount of time. Galaxy Zoo was incorporated for this very reason. Galaxy Zoo contains millions of images of galaxy and is open to the general public to classify the images of the galaxies. The galaxy zoo training dataset consists of 61,578 JPEG images of galaxies of size 424 * 424 with RGB channels, where each image has an ID number and galaxy at the centre. The training images of this challenge were given in a single directory and their respective classes were mapped in a .csv file with each image corresponding to 37 outputs. The test dataset in this challenge consists of 79,975 images of the galaxies without any probability labels. Therefore the task is to predict all the probabilities for the labels for the test dataset, and the result is evaluated by calculating the loss function over all predictions.

The decision tree considered presents the guidelines for measuring finer morphological features [21]. It stems from a citizen science crowdsourcing project where thousands of volunteers were asked to manually classify the galaxies based on 11 questions. There are 37 nodes or a total of 37 finer features that could characterise the galaxy. The output of the CNN's final layer gives the probabilities of all the 37 features. The task at hand was to identify the answers to all the 11 questions by determining the features having maximum probability under each class (question).

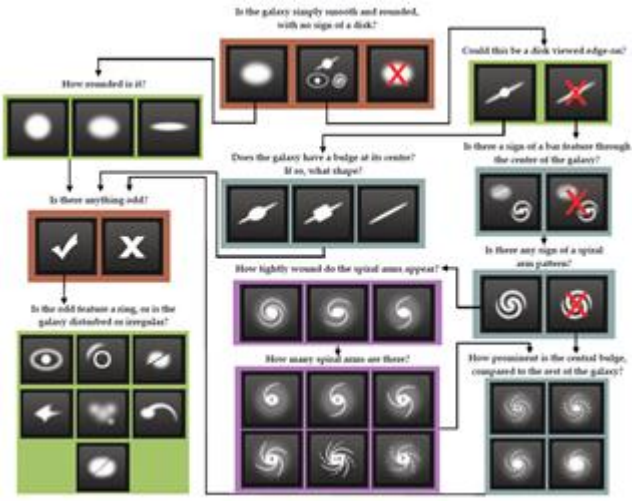

Fig 2: Galaxy Zoo Decision Tree (Willett et al.)

Once the strongest features are determined, they are presented as the physical features that characterise the input galaxy image that is uploaded by the user on the client-side interface. Each question's answer (the one with the maximum probability), determines the flow the tree. So the decision tree was coded as a function to which the data frame containing all the 37 answers 'probabilities was given as the input. The output of the function was a list of morphological features that characterise the galaxy as per the decision tree. The decision tree is shown in Fig 2.

III. PREPROCESSING

Image Preprocessing is very essential to analyse and make important inferences about celestial objects. This is done to improve the image data and suppress the undesired distortions.It plays a vital role in analysing and interpreting galaxy images and to improve and analyse the properties of celestial objects. Following are some Image processing techniques involved before feeding the image data into the models.

A. Resizing

Images were resized to 224x224 as the architecture was designed to accept images of the same size. Resizing also proves to be a vital step as it reduces the amount of computation otherwise involved. We used the PILLOW library of python to resize the images. An original image from the Galaxy Zoo dataset, after resizing, is shown in Fig 3.

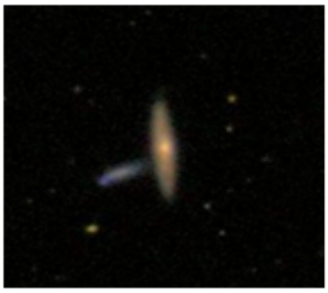

Fig 3: Resized image of the Galaxy Zoo Dataset

B. Data Augmentation

The performance of model usually improves with the amount of data available. And The main concerns of any model training is the problem of overfitting which can be overcome by using certain regularization Techniques.one among this is Data Augmentation. It is a technique of creating new data samples artificially from existing training data, thereby increasing the size of the training dataset. This is done by applying certain transformations to the images which includes random rotations, flipping, zooms etc. We employed the technique of rotation, where the images were rotated at random angles, thereby increasing the size of our dataset from 61578 to 246312 where each image was subjected to rotation in 4 random angles. We used PIL library of python to implement the same. We then applied the techniques of median filtering and contrast limited adaptive histogram equalization on the images.

C. Median Filtering

The median filtering technique was used to reduce noise in the images. It was chosen since it results in negligible loss of edges. The filter has a slight smoothing effect on our images. We employed the PIL library where ImageFilter module was used which has certain predefined filters that are used with the filter() function with kernel size set to 3. A median filtered image is shown in Fig 4.

Fig 4: Image after Median Filtering

D. Contrast Limited Adaptive Histogram Equalization

Generally, the contrast of the images can be increased by using a technique called histogram equalization. This technique, which increases the global contrast, sometimes does not yield better results and gives unclear images. We employed a variant method of adaptive histogram equalization called CLAHE (Contrast Limited Adaptive Histogram Equalization) which takes care of the over-amplification of contrast and avoids over-amplification of noise in relative regions of an image. CLAHE algorithm mainly operates on smaller regions in an image (sub-image) known as “tiles", rather than the entire image. Contrast limiting is used extensively to avoid the Noise. "Clip limit" is used to set the threshold for contrast limiting, when the histograms are above this specified threshold level, those pixels are clipped and distributed uniformly to other grey levels of an image. Therefore, it results in the prevention of over-amplification of contrast. LAB colour space is used,(Lab colour space is a 3-axis colour system with dimension L for lightness, A(red/green value) and B(blue/yellow value) for the colour dimensions). OpenCV was used to implement CLAHE. An image after applying CLAHE is shown in Fig 5.

IV. MODEL ARCHITECTURE

Our work made use of a Convolutional Neural Network for the classification process which was inspired by the VGG-16 architecture.

Fig 5: Image after CLAHE

CNNs are a class of Deep Learning algorithms that constitute convolution layers (where filters are made to scan the input to the respective layer) which play a vital role in the process of feature extraction. There are multiple architectures of CNNs used in related works. Our model was inspired by the VGG-16 architecture to which we made slight modifications. The VGG-16 architecture performed well in the ImageNet challenge, and it possesses a good depth of the network while retaining the structure’s simplicity. We made some slight modifications to the VGG-16 architecture. The network has 16 layers in total - 13 convolution layers and 3 fully connected layers. The original VGG-16 architecture makes use of a stride of 1 in the convolutional layer filters and a max-pooling layer with a stride of 2 which halves the dimensions. We, however, decided to go with a stride of 3 for the convolutional layer filters and a stride of 1 for the max-pooling layer while retaining the sizes to be 33 and 22 respectively for the two layers, the reasons being - a larger stride length leads to more generalisation and lesser overfitting which seems to be a major problem in the case of CNNs and image classification. We also chose a stride of 1 for the max-pooling layer to not reduce the dimensions (as that is already being done in the convolutional layers due to an increased stride) and highlight the strongest features only. The general approach is to choose a stride of 1 for the convolutional layer filters, which we also tried but we ended up getting a negligible change in the performance of the model in exchange for much larger computational requirements. The general architecture however, remained similar to the original VGG-16. The original architecture has around 138 million parameters, while our model had around 33 million trainable parameters, with almost the same level of performance, and was much faster to train. We used ReLu (Rectified Linear Unit) as our activation function in the convolution layers and a Sigmoid activation function in the output layer which contains 37 units in accordance to the 37 features as per the decision tree considered [21]. The architecture flow is shown in Fig 6.

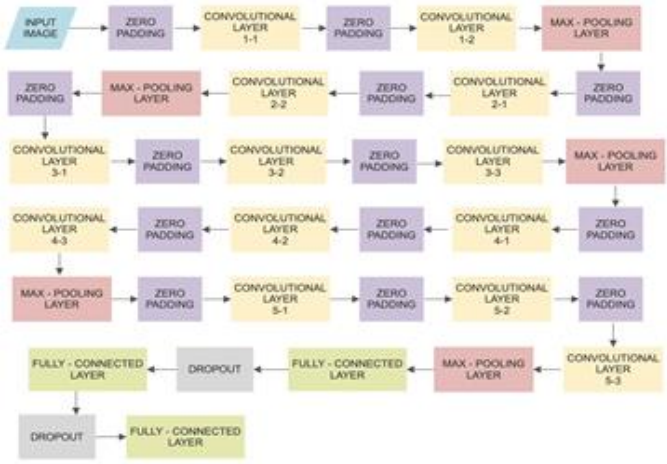

Fig 6: Model Architecture Flow

V. TRAINING

The first model that we trained was on unprocessed images. The unprocessed images were obtained after performing resizing, rotation and augmentation on original images from the dataset. We used RMSProp as optimizer with a learning rate e-6 and batch size 64. It was run for a total of 45 epochs. It yielded a training accuracy of 67.19% and a validation accuracy 67.98%. The training and validation loss in this case were the same that is 0.0162. We trained a second model on unprocessed images. We used RMSProp as optimizer but we changed the learning rate to e-4 and batch size to 32. It was run for a total of 50 epochs. It gave training accuracy of 82.22% and a validation accuracy of 75.46%. The training loss was around 0.0203 and validation loss was 0.0187. It per-

formed better than the first model. The third model was trained on processed images. Median filters and histogram equalization was done on unprocessed images to obtain processed images. We used RMSProp optimizer for this model. The learning rate was 10-4 and batchsize was chosen to be 32. It was run for 50 epochs and yielded a training accuracy of 81.87% and validation accuracy of 75.46%. The training loss was 0.0062 and validation loss was 0.0119. The fourth model was trained on original images from the dataset. The original images were not processed. They were resized, rotated, augmented by using the Keras built-in function called ImageDataGenerator() . The images were resized to 224 * 224 and were rotated and augmented using ImageDataGenerator(). The idea was to find out if the built-in function for data augmentation, rotation and resizing performed any better than the functions that we wrote from scratch to perform the data augmentation. It used Adam optimizer with a learning rate of 0.001 and the batch size was chosen to be 64. Decay was set to 5e-4. It ran for 12 epochs and gave a training accuracy of 59.97% and a validation accuracy of 61.96%. The training loss was 0.0203 and validation loss was 0.0187. The fifth model was also trained on original dataset images. The resizing, rotation and data augmentation was done using Image Data Generator(), built-in Keras function. We used Adam optimizer for this model. The learning rate was 0.001 and batch size was chosen to be 64. It was run for 15 epochs and yielded a training accuracy of 65.51% and validation accuracy of 62.78%. The training loss was 0.0119 and validation loss was 0.0172. The sixth model was trained on processed images. We used Adam optimizer for this model. The learning rate was 0.001 and batch size was chosen to be 64. It was run for 10 epochs and yielded a training accuracy of 72.96% and validation accuracy of 72.65%. The training loss was 0.0126 and validation loss was 0.0133. The seventh model was trained on processed images. We used Adam optimizer for this model. The learning rate was 3e-4 and batch size was chosen to be 64. It was run for 10 epochs and yielded a training accuracy of 77.15% and validation accuracy of 75.94%. The training loss was 0.0101 and validation loss was 0.0111. The eighth model was trained on processed images. We used Adam optimizer for this model. The learning rate was 0.001 and batch size was chosen to be 64. It was run for 13 epochs and yielded a training accuracy of 75.77% and validation accuracy of 74.54%. The training loss was 0.0108 and validation loss was 0.0120. The ninth model was trained on unprocessed images. We used Adam optimizer for this model. The learning rate was 0.001 and batch size was chosen to be 64. It was run for 6 epochs and yielded a training accuracy of 71.35% and validation accuracy of 70.76%. The training loss was 0.0108 and validation loss was 0.0150. This model yielded a relatively lower accuracy when compared to previous models. The final model was run in three different runs instead of one single run. Each run had varying number of epochs. The first run had 6 epochs and gave training accuracy of 74.31% and validation accuracy of 74.31% as well. The training loss and validation loss for first run was 0.0122. The second run was for 7 epochs and it gave training accuracy was 77.7% and validation accuracy was 76%. The training loss for the second run was 0.0099 and validation loss was 0.0110. The third run was for 9 epochs and gave a training accuracy of 78.56% and validation accuracy of 77.18%. The training loss for third run was 0.0088 and validation loss was around 0.0102.

Model	Dataset	Optimizer	Learning Rate	Decay	Batch Size	Epochs	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
1	Unprocessed Images	RMSPROP	0.00001	-	64	45	67.29%	67.98%	0.0162	0.0162
2	Unprocessed Images	RMSPROP	0.001	-	32	50	82.22%	75.77%	0.0059	0.0119
3	Processed Images	RMSPROP	0.001	-	32	50	81.87%	75.46%	0.0062	0.0119
4	Original Images	ADAM	0.001	0.005	64	12	59.97%	61.96%	0.0203	0.0187
5	Original Images	ADAM	0.001	0.005	64	15	65.51%	62.78%	0.0199	0.0172
6	Processed Images	ADAM	0.001	0.005	64	10	72.96%	72.65%	0.0126	0.0133
7	Processed Images	ADAM	0.003	0.001	64	10	77.15%	75.94%	0.0101	0.0111
8	Processed Images	ADAM	0.001	0.005	64	13	75.77%	74.54%	0.0108	0.0120
9	Unprocessed Images	ADAM	0.001	0.005	64	6	71.35%	70.76%	0.0141	0.0150

Table 1: Summary of the models trained

Run	Dataset	Optimizer	Learning Rate	Decay	Batch Size	Epochs	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
1	Processed Images	ADAM	0.001	0.005	64	6	74.31%	74.31%	0.0122	0.0122
2	Processed Images	ADAM	0.001	0.005	64	7	77.7%	76%	0.0099	0.0110
3	Processed Images	ADAM	0.001	0.005	64	9	78.56	77.18%	0.0088	0.0102

Table 2: Summary of the Final Model

This model which was trained for three different runs yielded a much better result than previous models which were trained in a single run. Table 1shows the summary of all the models trained.

VI. FINAL MODEL – PERFORMANCE AND RESULTS

The final model (i.e the tenth model) was run in three different runs instead of running the entire model in one single run. This model was trained on processed images, where the processed images were obtained by performing median filtering and histogram equalization on unprocessed images as discussed in previous sections .Each run had varying numbers of epochs and this model was built using Keras libraries and built in functions. We used Adam optimizer for this model. The learning rate was 0.001, decay was 5e−4 and batch size was chosen to be 64.

First run: The first run had 6 epochs. The loss plot has epochs as x-axis and loss as y-axis. As the the number of epochs gradually increased the loss gradually decreased and the accuracy increased as the number of epochs increased. The validation loss for first run was 0.0122. The validation accuracy at the end of first run was 74.31%.

Second Run: The second run was for 7 epochs. The loss decreased slightly as the epochs increased but validation loss became more constant towards the end of of 6th epoch. The validation loss at the end of second run was 0.0110 and the accuracy significantly increased until 4 epochs and then gradually increased as the number of epochs increased. The training loss for the second run was 0.0099 and validation loss was 0.0110. The validation accuracy at the end of second run was 76%.

Third Run: The third run was for 9 epochs. The loss decreased initially as the epochs increased but later remained mostly constant and started decreasing slightly at the end of 9th epoch which indicated overfitting. Therefore we stopped training after 9 epochs in the third run. and the validation ac-

curacy slightly increased during the first two epochs and then remained constant as the epochs increased. The validation loss at the end of third run was 0.0102.It gave a training accuracy of 78.56% and validation accuracy of 77.18%. Therefore ,This model which was trained for three different runs yielded a much better result than previous models which were trained in a single run. Hence this final model gave mean square error of 0.0102 on processed images. Table 2 shows the details of the final model. Fig 7 shows a sample output.

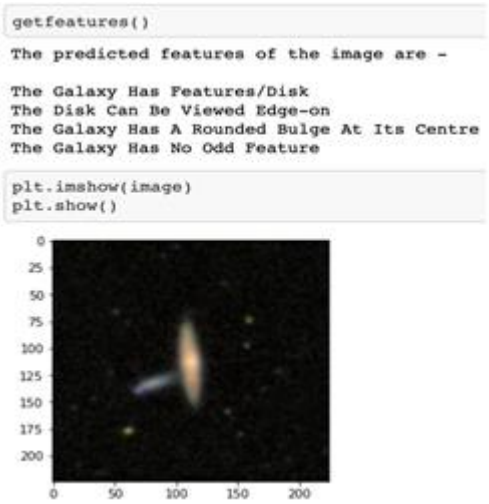

Fig 7: Sample Output

VII. MODEL DEPLOYMENT

Upon training the model, we deployed the model using flask wherein the user can upload an image and upon processing the image the results are rendered to the user. Flask was mainly used at the server side and at the front end we used HTML, CSS and Javascript. Two different HTML files are served to the browser one for uploading the image and the other for rendering the result to the user. At the backend we feed the image to the model and obtain the predicted features and render it to frontend. Http request-response protocol is implemented to communicate between the client (browser) and the server (flask).We created API endpoints to obtain user requests, based on the request obtained from the user, we sent appropriate responses. In the client-side interface, the user can upload an image of a galaxy and obtain its characteristic features as the output.

VIII. CONCLUSION AND FUTURE SCOPE

The deeper understanding of the dynamical history of galaxies is aided by the powerful probe of morphological parameters. Visual inspection for the identification of characteristic physical features of galaxies is impractical for astrophysicists. Our work presents a method of automating the process of determining the morphological features that characterise a galaxy with the use of Convolutional Neural Networks. The Galaxy Zoo dataset was resized and augmented, followed by the application of median filtering and contrast limited adaptive histogram equalisation. The processed images were then fed into a CNN based on the VGG-16 architecture. After the training of 10 models, the final model with the best combination of hyper parameters gave a mean square error of 0.0102 on processed images. The morphological features were then determined by the features having the maximum probabilities under each class or question. The model was then deployed onto a client-side interface using Flask where the user could upload an image of a galaxy and get a list of morphological features that characterise the galaxy as the output in real-time.

There are many secondary objects present in some images, so the scale can be reduced to see if it helps focus on the features of the galaxy under consideration, or segmentation algorithms for galaxies lacking distinct boundaries can be used. Many more data augmentation methods can be used. Techniques like flipping, translation, brightness/contrast changing etc. can be performed. Other architectures of CNNs like ResNet, AlexNet etc can be used to see if they perform any better. We have approached this problem as a multi-regression problem, but the probabilities of the features are not independent. Each question's answers apart from the first one, depends on a previous question's answer. We believe that utilising this constraint will lead to better results.

REFERENCES

[1]. M. Rahman, SN Azhari and Wahyono, "Classification of galaxy morphological image based on convolutional neural network", International Journal of Advanced Research in Science, Engineering and Technology, Vol. 5, Issue 6 , June 2018.

[2]. A. Gauthier, A. Jain, and E. Noordeh, "Galaxy Morphology Classification", Stanford University, Dated: December 16, 2016.

[3]. S. Kasivajhula, N. Raghavan, and H. Shah, "Morphological Galaxy Classification Using Machine Learning", Stanford University, 2007.

[4]. A. Kabani and M. El-Sakka, "How Important is Scale in Galaxy Image Classification?", Conference: International Conference on Computer Vision Theory and Applications, January 2016.

[5]. I. Dutta , S. Banerjee & M. De, "Shape Descriptors in Morphological Galaxy Classification", Advanced Computer Theory and Engineering (IJACTE), Volume-2, Issue-5, 2013.

[6].G.Tiwari, P.Mishal, T.Bhoye "Deepconvolutional neural networks for galaxy morphology classification", IRJET, Volume 6 Issue 3, March 2019.

[7]. P. H. Barchia, R. R. de Carvalho, R. R. Rosa, R. A. Sautter, M. Soares-Santos,B. A.D. Marques, E. Clua, T. S. Goncalves, C. de Sa Freitas, T. C. Moura"Machine and deep learning applied to galaxy morphology", Astronomy and Computing, 2019.

[8]. N. Eldeen. K.M. Hamed .N. Taha , A.E. Hassanien, I. M. Selim, "Deep galaxy V2: Robust deep convolutional neural networks for galaxy morphology classifications", The IEEE International Conference on Computing Sciences and Engineering (ICCSE), 2018.

[9].J.Dai,J.Tong,"Galaxy morphology classification with deep convolutional neural network", Astrophysics and Space Science, 2019.

[10]. A. Gauc, K.Z. Adam, J. Abela, "Machine learning for galaxy morphology classification", arXiv e-prints, 2010
[11]. H. Domnguez Sanchez,M. Huertas-Company, M. Bernardi, D.Tuccillo and J.L.Fischer,"Improving galaxy morphologies for SDSS with deep learning", Monthly Notices of the Royal Astronomical Society, 2018.
[12]. A. Martinazzo, M. Espadoto, Nina S. T. Hirata, "Self-supervised Learning for Astronomical Image Classification", 25th International Conference on Pattern Recognition (ICPR), 2020.
[13]. J. George, E. Kuminski, J. Wallin, L. Shamir, "Combining human and machine learning for morphological analysis of galaxy images", Publications of the Astronomical Society of the Pacific, 2014.
[14]. I.M.Selim,Arabi E. Keshk,Bassant M.E Shourbugy,"Galaxy image classification using Non-Negative matrix factorization", International Journal of Computer Applications, 2016.
[15]. D.Dhami, D. Leake, Sriraam Natarajan, "Knowledge-Based morphological classification of galaxies from vision features", AAAI Publications, Workshops at the Thirty-First AAAI Conference on Artificial Intelligence, 2017
[16]. Storrie-Lombardi, M.C.Lahav, O.Sodre, L., Jr., & Storrie-Lombardi, L. J.,Morphological "Classification of galaxies by Artificial Neural Networks", Monthly Notices of the Royal Astronomical Society, Volume 259, Issue 1, 1992.
[17]. J. de la Calleja, O.Fuentes, "Automated Classification of Galaxy Images",International Conference on Knowledge Based and Intelligent Information and Engineering Systems, 2004.
[18]. D. Misra , S. Mishra and B. Appasani, "Advanced Image Processing for Astronomical Images", Computer Science, ArXiv, 2018.
[19] A.K. Aniyan, K. Thorat, "Classifying Radio Galaxies with the Convolutional Neural Network", The Astrophysical Journal Supplement Series, Volume 230, Number 2, 2017.
[20]. J. Jenkinson, A. Grigoryan, M. Hajinorooz, R. Hern´andez, H. Barreto, A. Esquivel, L. Altamirano, V. Chavushyan, "Machine Learning and Image Processing in Astronomy with Sparse Data Sets", IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2014.
[21]. K. W. Willett, C. J. Lintott, S. P. Bamford, K. L. Masters, B. D. Simmons, K. R. V. Casteels, E. M. Edmondson, L. F. Fordson, S. Kaviraj, W. C. Keel, T Melvin, R. C. Nichol, M. J. Raddick, K. Schawinski, R. J Simpson, R. A. Skibba, A. M. Smith, D. Thomas, "Galaxy Zoo 2: detailed morphological classifications for 304,122 galaxies from the Sloan Digital Sky Survey", Monthly Notices of the Royal Astronomical Society, Volume 435, Issue 4, 11 November 2013.