Astronomy & Astrophysics manuscript no. output
March 11, 2022

Letter to the Editor

Classification of galaxies according to morphology and shape using CNN

Michalis Koumpanakis[1] and Giorgos Savathrakis[2] ⋆

[1] Physics Department, University of Crete, 71003, Heraklion, Greece
[2] Physics Department, University of Crete, 71003, Heraklion, Greece

Received 03 May 2020

ABSTRACT

With the rapid increase in the amount of data regarding the features of stellar and galactic objects, it is becoming increasingly harder to dedicate time to every object seperately in the attempt to classify it into a category. Therefore, it is becoming necessary to implement machine learning methods, in order to classify objects without having to examine new data ourselves. In this work, we will attempt to classify galaxies using photographic images, according to their shape and morphological features. This is to be implemented using a convolutional neural network whose input will be the images of the dataset and the output will be in the form of hard labels, which means that the output will be array-like and the non-zero value's position will determine the class in which each sample belongs.

1. Introduction

Each galaxy in the Universe has its own morphological features that enable its classification into distinct categories. These categories have to do with the shape, observational features e.g. spiral arms, and the way it is viewed from the observer. Traditionally, since the discovery of new galaxies by Edwin Hubble, their classification had to be done by the observers themselves. The initial categories in which galaxies were classified were ellipticals, spirals, barred spirals and irregulars. There are though more detailed features about each galaxy and these have to do on whether they are near-round or very elliptical, if they are ellipticals, the amount of spiral arms and how tight they are around the galactic center , if they are spirals and even if they are viewed edge-on or not.

As the classes become more detailed it is harder for individuals to do this classification work. Additionally, because of the ever increasing amount of data coming from all the space telescopes it is becoming near impossible to classify all this data by hand [Dai (2018)]. That is why a more computationally efficient and less time consuming method is required to classify astronomical objects and in our case specifically, galaxies. We are going to present a machine learning method that gives us the opportunity of taking an input image and classifying it into one of certain classes that will be defined in the next section. The method is based on using Convolutional Neural Networks (CNNs). A CNN is a deep learning algorithm that takes an image as input and learns information about it given the pixel values in every position of the image. For example, given an image, a CNN can classify it into a category by observing the correlation between each pixel and its neighbouring ones in the image. Since the data that we use correlate each image with its respective classes, we use architecturally different CNNs, we train the network on the images provided and then we evaluate its performance on a validation set. It is important to determine the way in which we decide which network had the optimal performance. Essentially, the network is trained in epochs and when the training is finished we get a final validation accuracy on the part of the dataset that is to be used as a validation set. After trying different forms of CNNs we get different validation accuracies. The optimal model is the one that yields the highest validation accuracy.

Another technique that is to be examined, is the use of pre-trained models in which the weights that connect the nodes and layers have been already determined by being trained on different datasets. It is not mandatory to use these models with the specified weights and what we are also going to examine is the efficiency of the model on our dataset using only its architecture. The pre-trained models that will be used are VGG16, VGG19, Xception and ResNet.

We present our code "Gal_Class.ipynb", written in python, which makes use of the tensorflow and keras libraries in order to apply the methods mentioned. Our data are from the Galaxy Zoo-the Galaxy Challenge[(GalaxyZoo 2013)], and they include a folder containing the images and a csv file connecting each figure to its respective properties that will link it to the classes that we are interested in. After implementing the methods forementioned, we are going to discuss the results by presenting the diagrams depicting the evolution of the validation accuracy over the epochs for each model that we create from scratch and each pre-trained model whether we include the weights or not. We are also implementing specific data manipulation methods that include resizing and cropping the images, which is necessary if we are to use the weights of pre-trained models and optional if we create a model from scratch, and we decide in which form the models give the best results. Finally, an important aspect of this work is to determine in which categories our data are to be classified and how using the probabilistic properties in which our data are described in the dataset, we transform them into hard-labeled classes.

2. Dataset

Our dataset is received from the kaggle competition Galaxy Zoo-the Galaxy Challenge. The entire dataset provides a folder containing 61578 training images, a folder containing 79975 test

⋆ *Present address:* Department of Physics, University of Crete, Heraklion, Crete, 71003, Greece

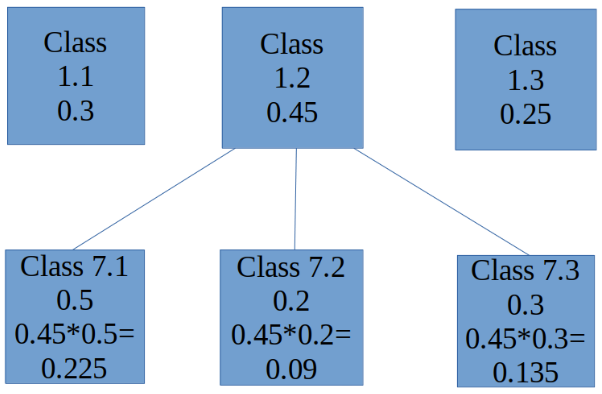

Class 1.1 0.3
Class 1.2 0.45
Class 1.3 0.25
Class 7.1 0.5 0.45*0.5= 0.225
Class 7.2 0.2 0.45*0.2= 0.09
Class 7.3 0.3 0.45*0.3= 0.135

Fig. 1. Example of connection between 2 clusters and how the probability of each class in the following clusters is calculated. If for example, class 1.2 were to be connected to cluster 7 then the probability of each class belonging to cluster 7 given that the object is definately belonging to class 1.2 would have to be multiplied by the probability of the object belonging to class 1.2 in order to obtain the final probability

images, a csv file containing the probability distributions for the classifications for each of the training images, and 3 additional csv files that have to do with standard points of reference for comparing the images of the dataset. These will not be used. The only data that we use are the images contained in the training folder and the csv file connecting each galaxy's id from the images' folder to its respective probability distributions.

The classification of the images is in the form of a tree. One cluster of classes comprises the root and according to which of the classes is the true one, the object is subsequently classified to a more specific type of class as shown in Figure 2. Each cluster of classes is connected to more clusters of classes according to the class that the object belongs similarly to the branches of a tree. The first cluster receives a probability for each of its classes. The selected class is the one that yields the highest probability. Since this is the selected class, the classification continues to the cluster that is immediately connected to it. For the following cluster there is another probability distribution for each of its classes. Therefore, the true probability of each class in this cluster is the propability of each class multiplied with the probability of the class of the previous cluster that connected to the current cluster as shown in Figure 1. These final probabilities, calculated in the same way that was explained, for each class in every cluster, are written in the csv file that connects each image with these probabilities. In our work however, we are only interested in classifying galactic objects to certain classes that do not reach the depth of the tree. The way we do that is explained in the following section.

3. Classification

3.1. Method 1

The number of classes that are to be used in this method are 9 and they are described in table 1. The main characteristic of this method is that it can include the entire dataset because every image can be categorised in one class. This is because the depth that is reached in the decision tree by selecting these specific classes gives a sum in the probability distribution equal to 1. Therefore, there has been no omission of classes that would likely include an object and since the decision on whether an object belongs to a certain class depends on which class yields the highest probability, we can be sure that not only do we keep the entire set but we are also sure that there has been no misclassification.

Table 1. Correlation between the class names that are used in method 1 and the respective ones from the dataset and their analytical description

Class	Dataset class	Description
1	Class 7.1	Completely round
2	Class 7.2	in-between
3	Class 7.3	cigar-shaped
4	Class 9.1	rounded bulge
5	Class 9.2	boxy bulge
6	Class 9.3	no bulge
7	Class 3.1	sign of a bar
8	Class 3.2	no sign of a bar
9	Class 1.3	not a galaxy (star)

Table 2. Classes used in method 2 with their description and the thresholds determining the placement of an object to a certain class using the probability distributions from the original dataset

Class	Description	Thresholds
0	Completely round smooth	Class1.1 ≥ 0.469
		Class7.1 ≥ 0.50
1	in-between smooth	Class1.1 ≥ 0.469
		Class7.2 ≥ 0.50
2	cigar-shaped smooth	Class1.1 ≥ 0.469
		Class7.3 ≥ 0.50
3	edge-on	Class1.2 ≥ 0.430
		Class2.1 ≥ 0.602
		Class1.2 ≥ 0.430
4	spiral	Class2.2 ≥ 0.715
		Class4.1 ≥ 0.619

3.2. Method 2

At this point we follow the classification method implemented by [(Dai 2018)], where 5 classes are selected and the decision rule for classifying an object into a class depends on a threshold which is placed in specific classes of the original dataset and the decision on whether or not an object belongs to a class depends on whether the object has a probability in each of the selected classes in the original dataset that exceeds the threshold. The classes selected in this method as well as the thresholds defined from the probabilities in the original dataset are presented in table 2. However, in this case we cannot use the entire dataset because there are objects that do not belong to any of these classes and they are subsequently dropped from the analysis.

4. CNNs

Convolutional Neural Networks(CNNs) are an advanced deep learning algorithm that is primarily used for image classification in the last few years.They were built to extract the most meaningful and important features of images.That is done by reducing the size of the input image to an appropriate size by keeping pixels that are more relevant to the classification problem.

CNNs have a unique architecture.A convolutional layer,in contrast to a simple DNN layer, consists of three main stages.In the first stage,multiple kernels/filters run over the image and per-

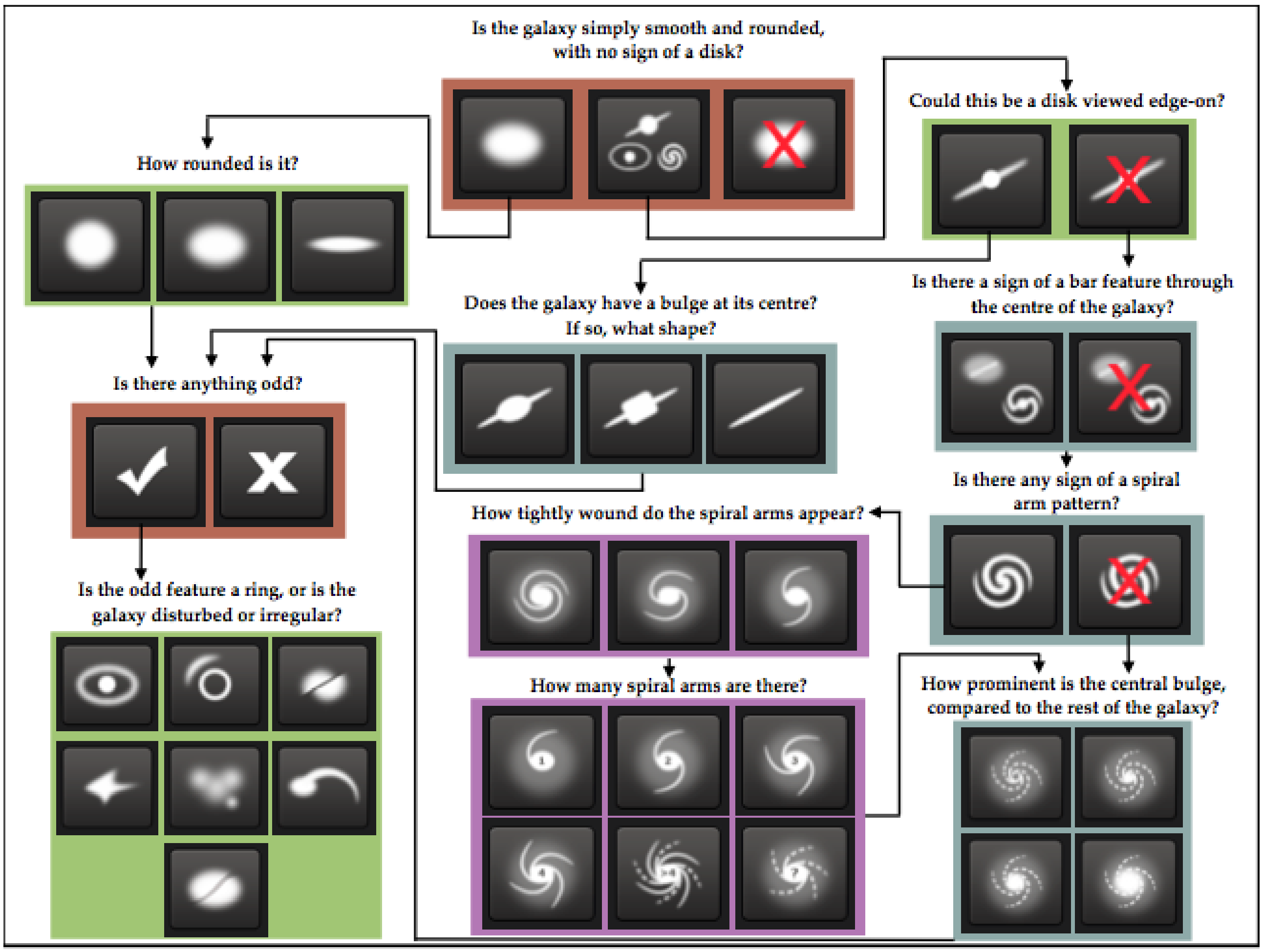

Is the galaxy simply smooth and rounded, with no sign of a disk?
Could this be a disk viewed edge-on?
How rounded is it?
Is there a sign of a bar feature through the centre of the galaxy?
Does the galaxy have a bulge at its centre? If so, what shape?
Is there anything odd?
Is there any sign of a spiral arm pattern?
How tightly wound do the spiral arms appear?
Is the odd feature a ring, or is the galaxy disturbed or irregular?
How many spiral arms are there?
How prominent is the central bulge, compared to the rest of the galaxy?

Fig. 2. Visualization of the decision tree that connects the classes. Each class belonging to a specific cluster can be connected to certain subsequent clusters. The final cluster is apparently not connected to any cluster but it is the last position where the previous clusters are connected

form a dot product.Let's say we have a $n \times n$ image .Kernels are usually 3×3 arrays that are initialized randomly.Their values are multiplied one by one with a 3×3 portion of the $n \times n$ image .The resulting value extracts information about that local part of the image.Then the kernel continues to run further on the image and produce new values.The resulting feature array is smaller in size from the original image and contains certain features of the image that depend on the kernel values.

In the second stage,a non-linear Activation Function is applied to the feature array that will determine which kernels to keep.The Rectified Linear Unit(ReLU) activation function is most commonly used in our case.

In the third stage,a max pooling layer is applied.The max pooling layer helps to further reduce the size of the image.Again a kernel runs on the feature array and keeps the largest pixel value from a specific region overlapped by the kernel.

Repeating the above enough times(with increasing number of neurons/kernels per layer) until the dimension of the image is small enough(e.g. 6×6) results in a complete convolutional algorithm.The previous convolution and pooling operations have greatly reduced the size of the input image to keep uniquely the meaningful characteristics for the classification.

The next step is to flatten the feature map(from 2-D) to a 1-D array.That array ,in turn,will be used in the training of classical DNN layers.Then a final output layer(softmax in our case) is used to predict the class of the image.

5. Pre-trained models

Part of this work is based on models that were already created and trained on different datasets. These models have their own convulutional neural network structure and after being trained on a specific dataset their weights are available for use. Their inclusion is optional but we are going to examine both cases and see which method gives the optimal results. In the analysis using pre-trained models we are going to use the 5 classes method explained in subsection 3.2. It is important to note that when the weights are included, the shape of the images in our dataset must be transformed to match the input shape of the pre-trained models. If they are not included then it is not necessary.

6. Analysis

6.1. Pre-trained models

In our work we include the pre-trained models VGG16[5], VGG19[5], Xception[6], ResNet50[3], ResNet152[3] received from the keras library. The analysis of the results obtained using these models is split into 2 parts. The results obtained when the weights that were trained on imagenet are included, and the results when they were not included.The labels used were derived from method 2(predictions on 5 classes).The total number of images that were used before the conversion into hard labels were 30000.After the 5 classes are derived based on the confi-

Table 3. Pre-trained models' input shapes when the imagenet weights are included and not

Model	Input Shape (with weights)	Input Shape (without
VGG16	(224,224,3)	(106,106,3)
VGG19	(224,224,3)	(106,106,3)
Xception	(299,299,3)	(106,106,3)
ResNet50	-	(106,106,3)
ResNet152	(224,224,3)	-

Table 4. Validation accuracies for each pre-trained model when the imagenet weights are included

Model	Validation Accuracy
VGG16	0.7400
VGG19	0.6650
Xception	0.6879
ResNet152	0.3707

dence intervals from Table 2 a total of 13589 images remain for our analysis(some images don't belong to any of these classes).

6.1.1. Included Weights

At this point the training images had to be resized to the input shape of the pre-trained models. The input shapes required for each of the pre-trained models, when the weights from imagenet are included, are presented in table 3. The objects belonging to the classes that we selected for our classification task, are split in a train to test proportion 90/10.All convolutional layers are received from the pretrained models.Two hidden DNN layers along with an output layer(softmax) were added to the models.The loss function used was 'categorical loss entropy' and the optimizer was ADAM .The models are trained for 20 epochs and the batch size for each training step is 32. The models used are VGG16, VGG19, Xception and ResNet152. The results regarding the accuracies and losses are presented in figures 4 and 5 respectively. We observe that the highest validation accuracy is achieved by model VGG16. However, it should also be noticed that the least overfitting model is Xception though its performance is slightly worse. This is observed because the validation loss for Xception seems to drop gradually over every epoch as opposed to all the other models used in this analysis. We present the validation accuracies for each of the pre-trained models in table 4. The results concerning the accuracies on the validation sets are not encouraging since the maximum accuracy is just over 70%. So it is important to check the performance of these models when the weights are not included.

6.1.2. No weights

The size of the images when the weights are not included is manually processed. The transformation of our images was done in the following way. Firstly, the images were resized to 224×224 from the original 424×424 size, then center cropped to 120×120 and finally resized again to 106×106. This is shown in table 3.Then we rescale the pixels to the [0,1] interval(normalization) by dividing all pixel values with 255.This time we trained our models for 10 epochs keeping the same train/test ratio and using the models VGG16, VGG19, Xception and ResNet50(with the same model architecture described in (6.1.1)). The results in the accuracies and losses are shown in figures 6 and 7. An apparent prevail in the validation accuracy is observed for the

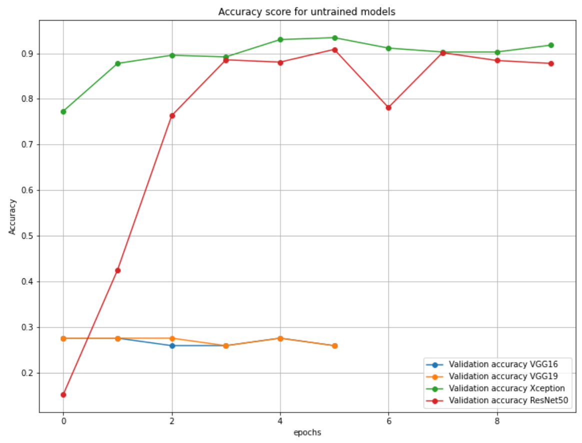


Fig. 3. Combined validation accuracies for all pre-trained models without including the weights

Table 5. Validation accuracies for each pre-trained model when the imagenet weights are not included

Model	Validation Accuracy
VGG16	0.2600
VGG19	0.2600
Xception	0.9179
ResNet50	0.8779

Xception model where it exceeds 90%. The combined validation accuraccies for all the models are shown in figure 3. It is also observed that ResNet50 may also be a good candidate as a preferable model for our classification task and it is possible that given more epochs of training it might as well exceed Xception at some point. The validation accuracies in the final epoch for each of these models are presented in table 5.

6.2. Main model

After testing the above models , we tried building our own model to see if it's possible to achieve better predictions.We used both method 1 and method 2 from section 3 to create our hard-labels and make predictions.As we will see method 2 was much more effective.

6.2.1. Preprocessing

Like in section 6.1.2 the original images have a size of 424×424.Because it's difficult to rescale images with such many pixels they are resized to 224×224 then center cropped to 120×120 and finally resized to 106×106.For method 1, were 9 classes were chosen for labels according to the highest probability ,30000 images are used for analysis while for method 2 only 13589.After the train/test split of 0.9/0.1 is applied to the data,the images are rescaled to the [0,1] interval(pixel values divided by 255).

6.2.2. Network architecture

The model consists of 10 convolutional layers with increasing number of neurons every 2 layers(e.g. [32,32,64,64,...,512,512]),

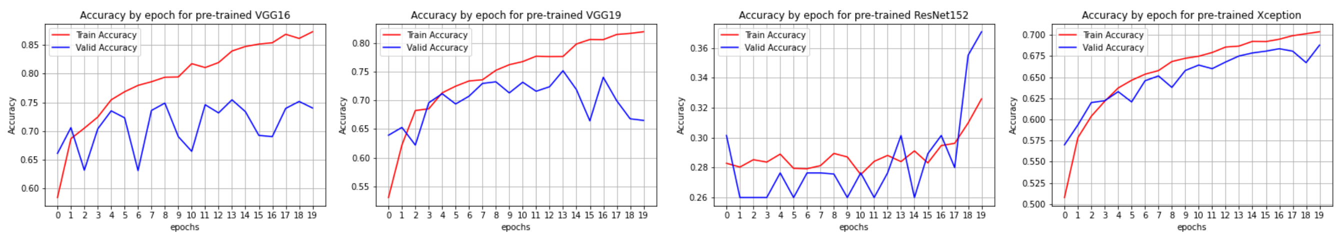


Fig. 4. Accuracy scores for the validation and train sets over 20 epochs for VGG16, VGG19, ResNet152, Xception. The weights of the models' training on imagenet are included.

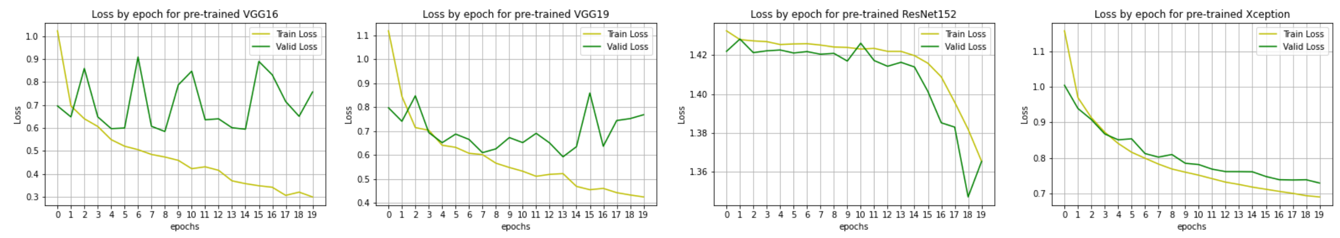


Fig. 5. Loss for the validation and train sets over 20 epochs for VGG16, VGG19, ResNet152, Xception. The weights of the models' training on imagenet are included.

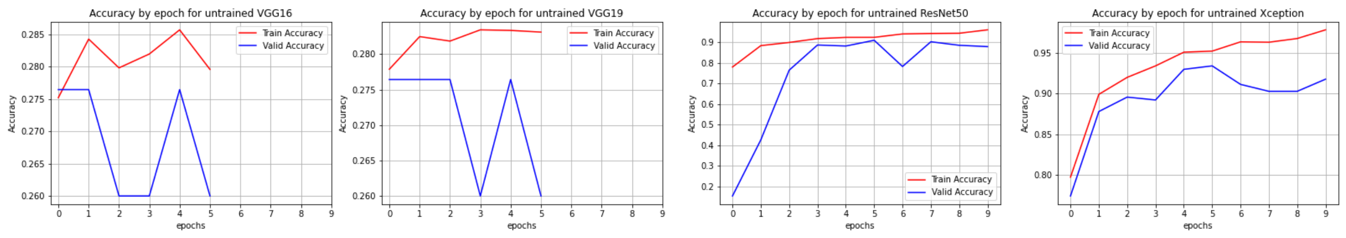


Fig. 6. Accuracy scores for the validation and train sets over 10 epochs for VGG16, VGG19, ResNet50, Xception. The weights of the models' training on imagenet are not included.

3 hidden layers [512,256,128] and a softmax activation function with 9 neurons as an output for method 1 and 5 neurons for method 2.After each layer a Leaky ReLU activation function is applied.

ReLU is a piecewise linear function that pruns the negative part to zero and retains the positive part. Under such constraint, there is a possibility for ReLU neurons to become inactive when you have values only smaller than 0. Leaky ReLU solves this problem by not pruning the negative part to zero with a slope alpha depending on the method we are using.

After the activation function is applied we perform Batch Normalization.Batch Normalization[4,BN (2015)] is a technique that normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.Likewise in section 6.1.2 we normalize our input images to the [0,1] interval to speed up training.The same applies here for each layer to make sure training doesn't get stuck because of low and high values in activation functions.

A max pooling layer is applied(after 2 convolutional layers) that reduces the size of images by half(with a 2×2 kernel and a stride of 2).Finally a dropout layer is added to regularize the model in order to avoid overfitting.

Dropout[8] randomly switches off some neurons in a layer so that they do not contribute any information or learn any information(for each sample) during training.Although training becomes slower due to a reduced number of neurons that are being used, an increased performance of our model in the test set is ensured.

Dropout and max pooling are applied after 2 convolutional layers have been used,in contrast to Batch Normalization and Leaky ReLu which are applied after each layer.

6.2.3. Implementation

Each convolutional layer has a kernel initializer(he_uniform) as in [3, He-et-al. (2015)] and a padding of 'same' which ensures the size of images stays the same after the convolution operation is applied.Before the final feature array is flattened its size is 3×3.

The optimization algorithm is ADAM[7] with a loss of 'categorical loss entropy'.The model is trained for 30 epochs and the batch size for each training step is 32.The images are shuffled for training and we use the test data as validation due to the shortage of images.All of the above are implemented for both methods.

6.2.4. Method 1

In this method the dropout rate is [0.2,0.3,0.4,0.4,0.5,0.5,0.4,0.3] and is implemented every 2 layers for the convolutional ones and every layer for the dense ones.The value of Leaky ReLU's slope a is 0.01 for every layer.

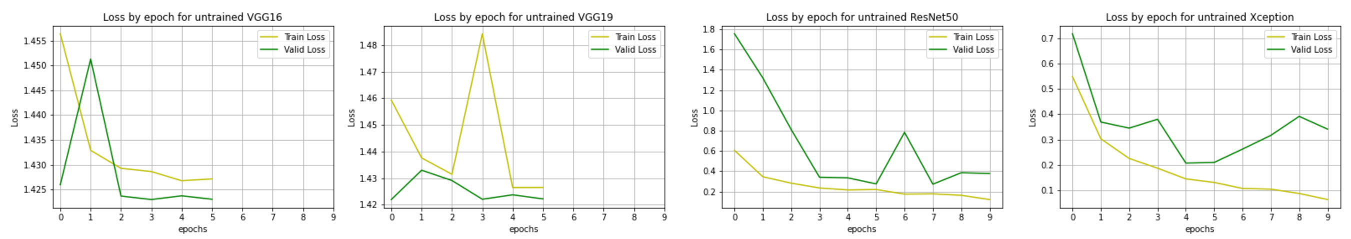


Fig. 7. Loss for the validation and train sets over 10 epochs for VGG16, VGG19, ResNet50, Xception. The weights of the models' training on imagenet are not included.

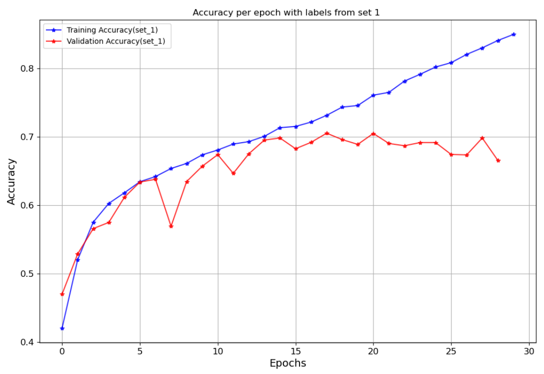


Fig. 8. Accuracy scores for the validation and train sets over 30 epochs(main model method 1)

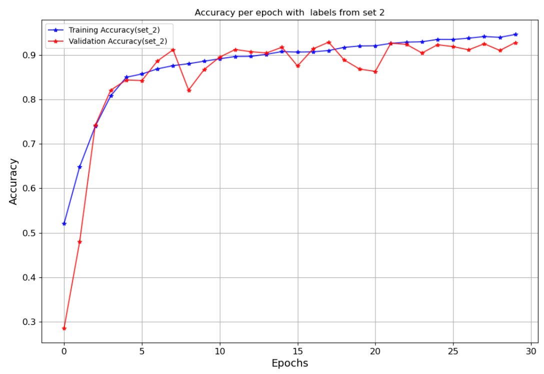


Fig. 9. Accuracy scores for the validation and train sets over 30 epochs(main model method 2)

6.2.5. Method 2

Likewise the dropout rate in method 2 is repeated with a slight change for the 256-neurons layers that have a value of 0.45.Leaky ReLU's slope a is 0.1 for every layer.

7. Results

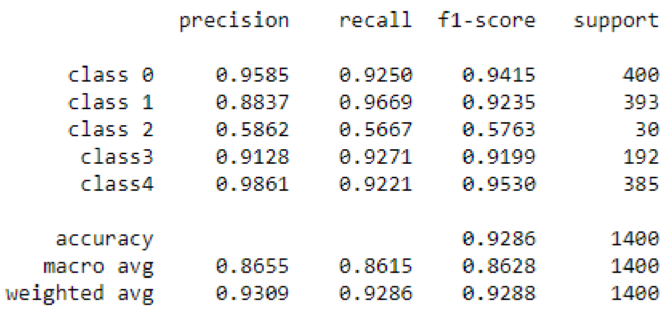
7.1. Method 1

In our first approach to the model were we chose to keep the highest probabilities for 9 classes in order to convert the soft-labels to hard-labels the results were underwhelming.The model had a maximum accuracy of 0.7053 and it started to overfit after 15 epochs even with dropout.A visual representation of the results can be seen at figure 8.

The main problem of this approach was that even with 30000 images many classes had much less images than other classes,making training for all classes difficult.For example class 5 and 9 had only 50 training images each while class 3 had over 12000 images.Furthermore some classes were very closely related.During the conversion from soft to hard labels some classes had very close probabilities with other classes(e.g. 0.48,0.52).This problem is solved in method 2 were the selection of classes is based on a threshhold[table 2] for each class that depends on each branch of the galaxy zoo tree.For this reason no further analysis will be covered for this method as it is flawed.

7.2. Method 2

In our second approach with 5 classes the results were much more satisfying.The maximum overall validation/test accuracy was 0.9286 and was recorded in epoch 30.With only 13589 training images we managed to rapidly increase the accuracy of our model within reasonable epochs(20 minutes to train).A possible performance improvement can be achieved if data augmentation technique is used like in [2] in order to further increase the number of images for training.



```
              precision    recall  f1-score   support

     class 0     0.9585    0.9250    0.9415       400
     class 1     0.8837    0.9669    0.9235       393
     class 2     0.5862    0.5667    0.5763        30
      class3     0.9128    0.9271    0.9199       192
      class4     0.9861    0.9221    0.9530       385

    accuracy                         0.9286      1400
   macro avg     0.8655    0.8615    0.8628      1400
weighted avg     0.9309    0.9286    0.9288      1400
```

Fig. 10. Classification report

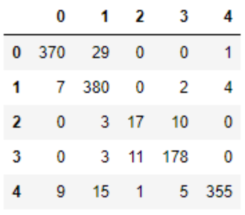
7.2.1. Performance metrics

A deeper view of the model's performance can be seen in Fig.10 where the accuracy for each class is showcased.The precision,recall and f1-score metrics are defined as:

$$P = \frac{TruePositive}{TruePositive + FalsePositive} \tag{1}$$

$$R = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2}$$

$$f_1 = \frac{2PR}{P + R} \tag{3}$$



	0	1	2	3	4
0	370	29	0	0	1
1	7	380	0	2	4
2	0	3	17	10	0
3	0	3	11	178	0
4	9	15	1	5	355

Fig. 11. Confusion matrix of our model for each class on testing set.Column represents true label and row represents prediction label.

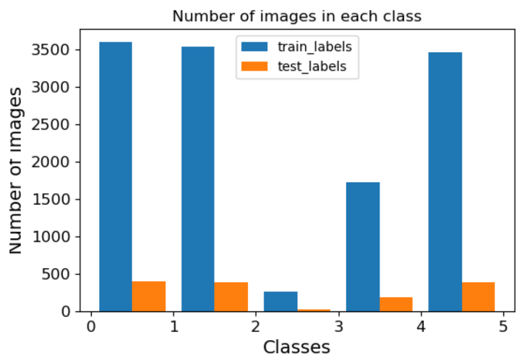


Fig. 12. Distribution of images per class.

Furthermore the confusion matrix can be seen at Figure 11.The accuracy of the 5 galaxy types are: completely round 92.5%, in-between 96.69%,cigar-shaped 56.67%, edge-on 92.71% and spiral 92.21% respectively.As we can see images belonging in the third class[cigar-shaped smooth] are easily misclassified into other classes.The main reason this happens is due to the shortage of images belonging to this class(showcased in Figure 12) .Completely round galaxies(class 0) are misclassified as in-between(class1) and vice versa ,something to be expected due to their similarity.Class 5 is also misclassified into all other 4 classes with a lower frequency as it is hard to distinguish spiral arms in some of the images.

7.2.2. Distribution of images

A visual representation of the number of images belonging to each class both for the test and training set can be seen at Figure 12.The third class has a very limited number of images for training,making predictions for this class harder.

8. Conclusions

Our model achieves the best classification performance, the overall accuracy on testing set is 92.86% beating Xception's 91.79% score. Even though Xception was only trained for 10 epochs and relatively fast its performance was really good.Given more data it may be able to beat our own model(Xception was originally built to train on very large datasets e.g. imagenet).The in-between round-smooth galaxies were the ones with the highest test accuracy(96.69%) while the cigar-shaped ,the ones with the lowest(56.67%). The selection of labels was very important.As we can see method 2 outperforms method's 1 performance by 31% even though we train on half the images.Pre-trained models like VGG16 performed well with an accuracy of 74%.If we take into account that we only train the non-convolutional hidden layers and training is much faster then the trade of accuracy for speed is sufficient.Finally we can conclude that the classification of galaxies with the use of CNNs is successful and can be used to automate the process of classification for astronomers.

References

1) Dai, J., Tong, J., 2018 arXiv preprint arXiv:1807.10406

2) Kyle W. Willett, Chris J. Lintott.., 2013, Galaxy Zoo arXiv, preprint arXiv:1308.3496

3) Learning for Image Recognition(2015) Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2015 , Deep Residual Learning for Image Recognition ,arXiv preprint arXiv:1512.03385

4) Sergey Ioffe, Christian Szegedy, 2015 , Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift ,arXiv preprint arXiv:1502.03167

5) Karen Simonyan, Andrew Zisserman,2014,Very Deep Convolutional Networks for Large-Scale Image Recognition,arXiv preprint arXiv:1409.1556

6) François Chollet, 2014 , Xception: Deep Learning with Depthwise Separable Convolutions, arXiv preprint arXiv:1610.02357

7) Diederik P. Kingma, Jimmy Ba , 2015 , Adam: A Method for Stochastic Optimization,arXiv preprint arXiv:1412.6980v9

8) Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, Ruslan R. Salakhutdinov, 2012 , Improving neural networks by preventing co-adaptation of feature detectors , arXiv preprint arXiv:1207.0580