**Galaxy Classification inspired by SqueezeNet and MobileNet :**

**Combining efficiency and accuracy**

1. **INTRODUCTION**

Astronomer Edwin Hubble led the way to classify celestial giants called galaxies. Galaxies are a system of stars, interstellar gas, dark matter, dust, and stellar remnants that are gravitationally bound together. Galaxies form over billions of years, and their morphology – essentially their shape and general visual appearance – gives astronomers much information about their composition and their evolution.

Galaxy classification is important because astrophysicists frequently make use of large catalogs of information to test existing theories against or to form new conjectures to explain the physical processes governing galaxies, star formation, and the nature of the universe. Currently, astronomers manually classify galaxies based on visual inspection of photographs. This method is slow and is certainly not a worthy activity for an astronomer to be engaged in. This method is also prone to human error and thus accounts for some inaccuracies and misclassifications. Astronomy has recently seen an explosion of data,due to programs like the Sloan Digital Sky Survey (SDSS) .

The Sloan Digital Sky Survey (SDSS), which started in 2000, collected more data in its first few weeks than had been amassed in the history of astronomy. Now, 20 years later, its Article published by EDP Sciences A122, archive contains about 170 terabytes of information. Soon its successor, the Large Synoptic Survey Telescope (LSST), will acquire that quantity of data every five days (York et al. 2000). It provided entry points for the computer scientists who want to engage in astronomical research, and explains why big data mining and machine learning methods are gaining such popularity.

Since access to this amount of data has only become possible in the past decade, computer-aided celestial classification is a very young area, with much scope for machine learning and image processing application. Our goal is to apply machine learning algorithms to the repetitive task of galaxy classification on a massive data set. This will not only decrease classification errors but will also allow astronomers to pursue more stimulating tasks. At the same time, we have tried to deploy such models which have less no of parameters and can be used for mobile and web applications.

**1.1 Convolutional Neural Networks**

Most computer vision algorithms use something called a convolution neural network, or CNN. A CNN is a model used in machine learning to extract features, like texture and edges, from spatial data.

Like basic feedforward neural networks, CNNs learn from inputs, adjusting their parameters (weights and biases) to make an accurate prediction. However, what makes CNNs special is their ability to extract features from images.CNNs are able to treat images like matrices as they exist and extract spatial features from them, like texture, edges and depth. They do this by using convolutional layers and pooling.We can consider convolutional layers as a set of feature maps — the convolutional layer applies a series of image filters to an input image represented as a matrix. The resulting filter images, or feature maps, have different appearances, as they extracted different features.These image filters are called convolutional kernels.

1. **Review of State-of-Art and Related Works**

In the context of Classification and Morphology of Galaxies, we review below several works where different approaches were developed and great efforts were made to identify the morphological types of galaxies from the SDSS in the visual and in the automated modes..

Calleja et. al. presented an experimental study for using machine learning and image analysis in order to classify galaxies. A neural network with a locally weighted regression method, and homogenous ensembles of classifiers were used. They used the bagging ensemble method for the neural networks and they manipulated input features to create the ensemble of locally weighted regression. The galaxies were transformed by rotating, and center-cropping, in a fully automatic manner. Furthermore, Principal Component Analysis (PCA) was used for dimensionality reduction, and to extract relevant information from the image data. The preliminary experimental results were evaluated with a ten-fold cross validation technique, and it showed that the homogenous ensemble of locally weighted regression produces the best results, with 91 percent accuracy when considering three types of galaxies (Elliptical, Spiral, and Irregular), and 95 percent when considering two types (Elliptical and Spiral).

Kasivajhula et al. (2007) explored support-vector machine, random forest, and naive Bayes algorithms as the galaxy image classifiers, and principal component analysis for the direct image pixel data compressing, but favored random forest. Nevertheless, Andrae et al. (2010) applied a probabilistic classification atenfold to classify the SDSS bright galaxies and obtained that it produces reasonable morphological classes and object-to-class assignments without any prior assumptions.

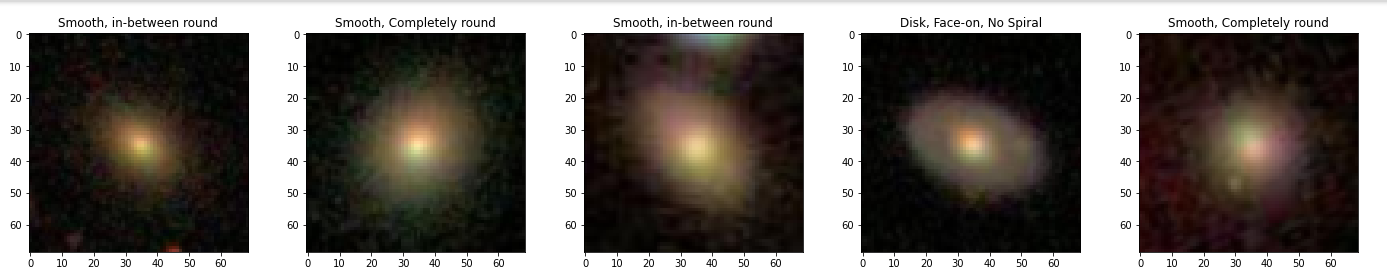
Chou et. al. set out to build an algorithm that can extract indicators for galaxy morphologies. A pipeline was developed which combined multiple computer vision feature detectors and ML regression. The performance was experimented using the cross-validation technique. There are 3 sections in the pipeline: feature extraction, machine learning regression, and probability normalization. Multiple techniques were used for image analysis, like PCA, SIFT, Hog, Fourier transforms, etc. Their neural network is pre-trained, called Overfeat, which was trained on the ImageNet dataset

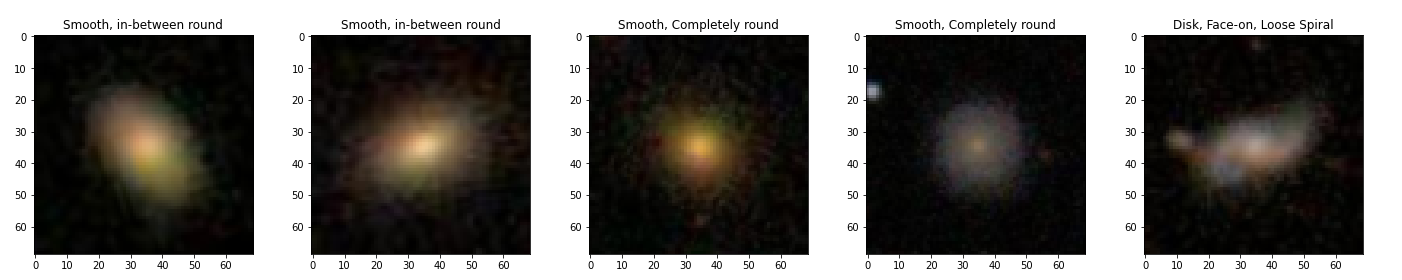
Gonzalez et. al. presented a method to automatically detect and classify galaxies which include a novel augmentation procedure to make trained models more robust against the data taken from different instruments and contrast stretching functions. Training of the deep learning models was done using the public data such as the SDSS and Galaxy Zoo dataset, and private ones such as the Next Generation Virgo (NGVS) and Fornax (NGFS) surveys. Training were strongly bound to the conversion method from raw FITS data into a 3 channel RGB image. Therefore, a proposal for using 5 conversion methods in data augmentation. This resulted in great improvement in the overall detection of galaxies from different instruments, data reduction procedure, and bands. The deep learning framework DARKNET and YOLO real-time object detection system were used to train the detection and classification methods. They were implemented in C and the CUDA platform, making extensive use of graphical processing units (GPU), which could process an SDSS image in 50 ms, or a DECam image in around 3 seconds.

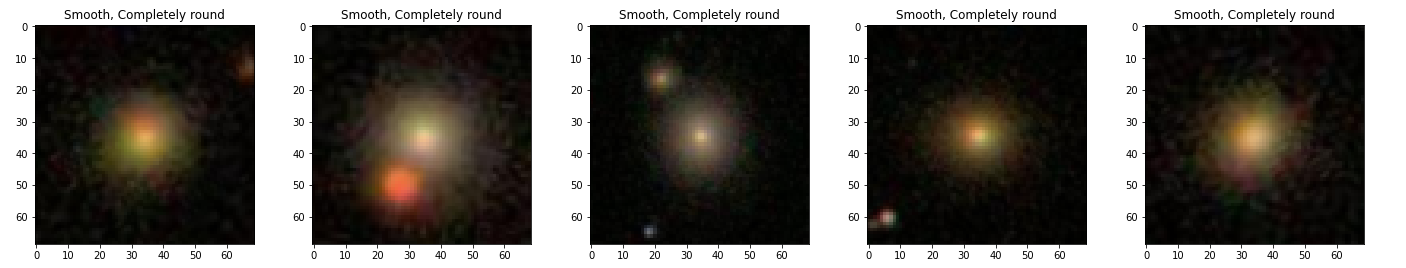
**DATASET**

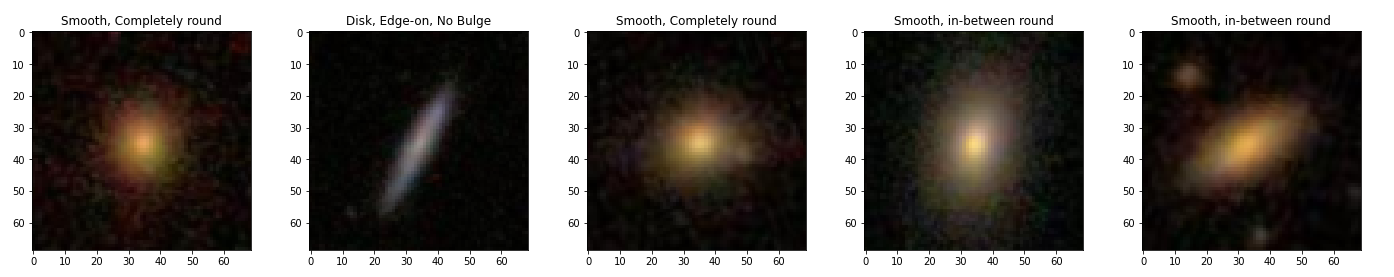
For our study, we used AstroNN’s SDSS dataset which contains 21785 69x69 pixels colored galaxy images separated in 10 classes. The name of the classes along with their example images is showcased in fig1.

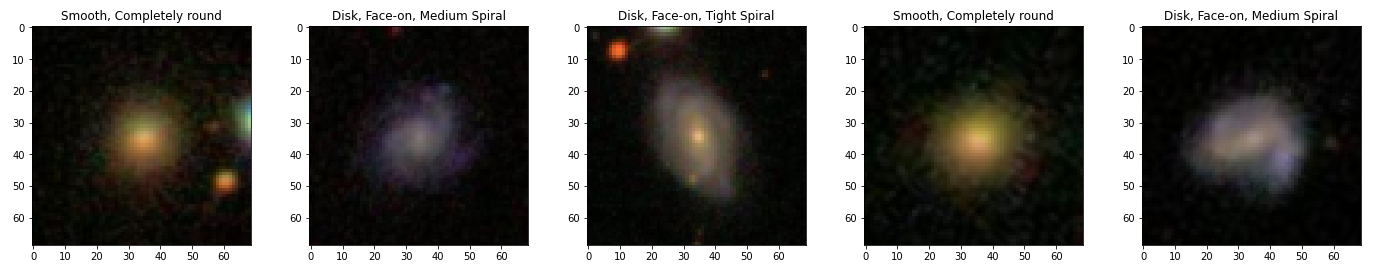
**Fig1.**

****

****

****

****

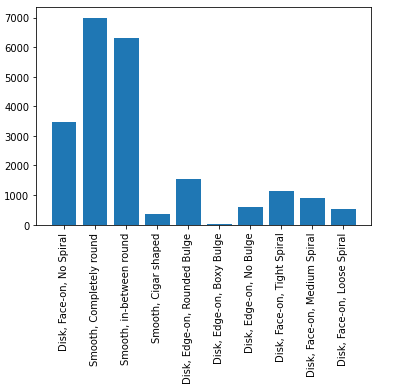
****

All the label information for this dataset was taken from GalaxyZoo. Galaxy Zoo is a project that relies on volunteers to classify galaxy images, and as a result, there can be disagreements among the volunteers. To address this, Galaxy10 only includes images where more than 55% of the votes agree on the classification, resulting in a dataset of 21,785 images. This threshold was chosen based on validation and is meant to be an alternative to MNIST or Cifar10 as a deep learning toy dataset for astronomers. The images were cropped and downscaled to make them manageable on most computer and graphics card memory

The class distribution of these images is given in Table2.

Table 2.

|  |  |  |
| --- | --- | --- |
| Class No. | Class Name | No. of Images |
| 1 | Disk, Face-on, No spiral | 3461 |
| 2 | Smooth, Completely round | 6997 |
| 3 | Smooth, In-between round | 6292 |
| 4 | Smooth, Cigar shaped | 394 |
| 5 | Disk, Edge-on, Rounded bulge | 1534 |
| 6 | Disk, Edge-on, Boxy bulge | 17 |
| 7 | Disk, Edge-on, No bulge | 589 |
| 8 | Disk, Face-on, Tight spiral | 1121 |

****

1. **Proposed Solution(s)**

In order to make our work application friendly, we tried solving our problem statement with the aim of making our model as smaller as possible. Therefore, we implemented small CNN’s namely Squeezenet and MobileNet which gives Alex-net level accuracy with fewer parameters.

**Squeezenet**

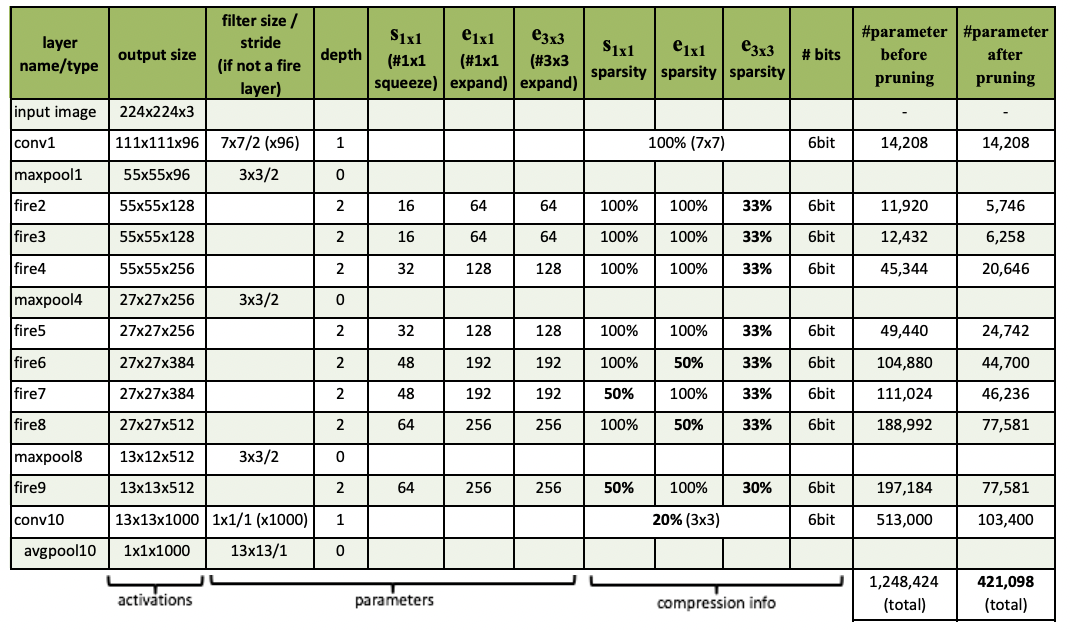
SqueezeNet is a neural network architecture designed for efficient deep learning on constrained computational resources, such as mobile devices or embedded systems. It achieves this by using a combination of techniques such as network pruning, reduction in the number of input channels, and the use of 1x1 convolutional filters to reduce the number of parameters and operations required.

The core of the SqueezeNet architecture is the Fire module, which consists of a squeeze layer that uses 1x1 convolutions to reduce the number of input channels, followed by an expand layer that uses a combination of 1x1 and 3x3 convolutions to generate a set of output feature maps. The Fire module is repeated multiple times to form the overall network.

SqueezeNet also uses techniques such as global average pooling and softmax regression for classification, which further reduce the number of parameters in the network. Overall, SqueezeNet achieves state-of-the-art accuracy on several image classification benchmarks while being much more computationally efficient than other popular neural network architectures.

SqueezeNet begins with a standalone convolution layer, followed by 8 Fire modules and then ending with a final conv layer. There is a gradual increase in the number of filters per fire module from the beginning to the end of the network. SqueezeNet performs max-pooling with a stride of 2 after layers conv1, fire4, fire8, and conv10; these relatively late placements of pooling are per Squeezenet’s Strategy. The full SqueezeNet architecture is presented in Table 1.

**Table 1.**



**Methodology**

* **Data loading and pre-processing**

For our study, we have used the galaxy10 dataset from a python module called AstroNN. Images and Labels are loaded from the respective module where further processing like normalizing the images and hot encoding the labels is done. We further split the data into training and testing at a ratio of 85 : 15.

* **Architecture construction**

In our model, the fire module is the most crucial component.

The Fire module has been widely adopted in various deep learning architectures, not just in SqueezeNet, but also in other models such as MobileNet and EfficientNet. Its ability to reduce model size and computational requirements has made it a popular choice for edge devices and other resource-constrained platforms. The structure of the fire module is shown in Fig2.

We have used convolution layers to build this fire module. The activation function “relu” is used in these convolution layers. Rectified Linear Unit (relu) function returns 0 if it receives any negative input, but for any positive value it returns that value back. So it can be written as :

f(x)=max(0,x).

The filter is applied using convolution layers to produce a feature map.

Each 2D convolution layer uses a n x n array and a m x m kernel to apply k

filters and retrieve k features. Here, the input image's (individually the r, g, and b bands) dimensions are n x n, the convolution kernel's dimensions are m x m,

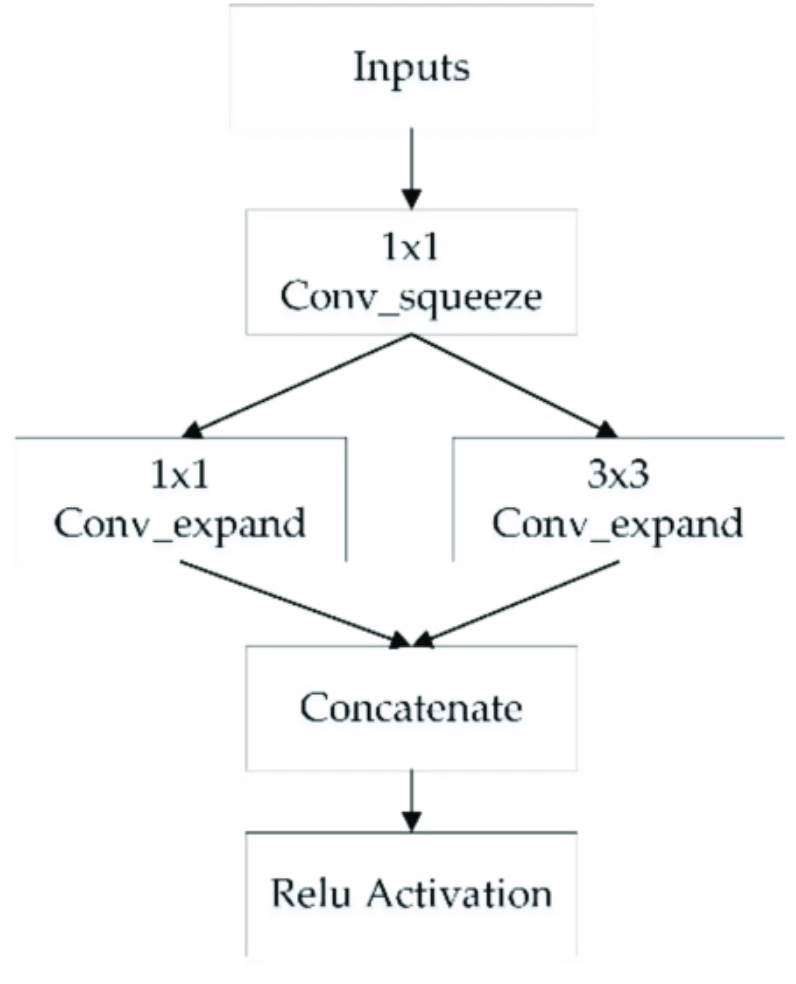
and the number of filters is k.

Our model architecture has multiple fire module blocks followed by a few Maxpooling layers in between those fire blocks. Similarly, Global average pooling is applied at the end of our architecture.

We employ pooling layers with convolution layers to reduce the feature map's sample size. In this study, MaxPooling was employed to assist keep the maximum components and lessen the noise and GlobalAverage pooling was employed to generate one feature map for each corresponding category of the classification task in the last convolution layer.

"Softmax" serves as our activation function in the output layer. This is done so that the model's output will be the likelihood that a picture belongs to a certain class.

This implementation of Squeezenet architecture was done using Keras tensorflow.

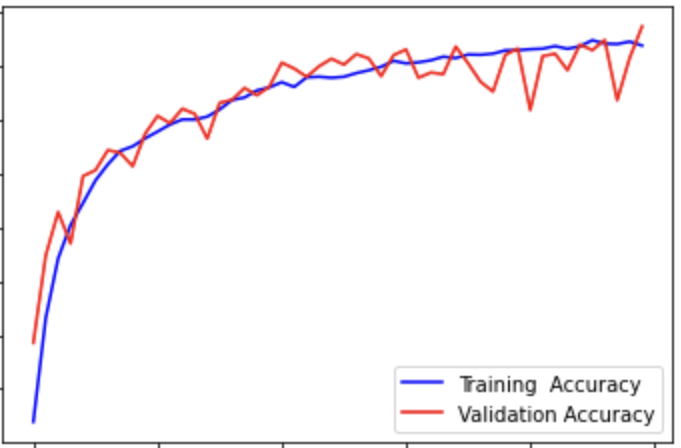


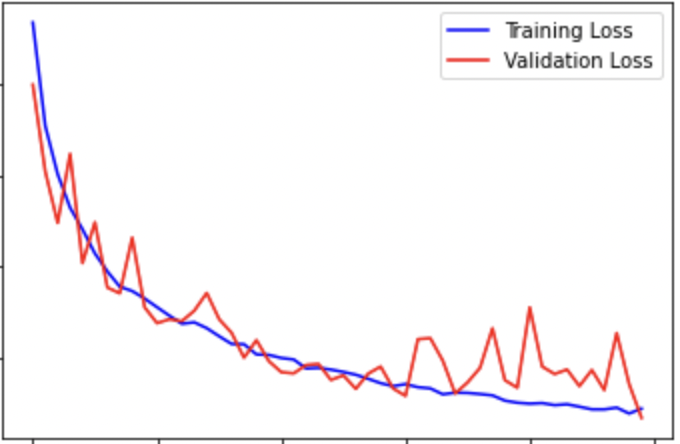
**Fig 2.** Structure of the fire module

* **Model training**

As we split the data into training and testing at the ratio of 85 : 15, the model training was done on 18517 images and testing on 3268 images respectively. The total number of parameters shown in the model summary were 740,554. While training, Categorical cross-entropy loss was used as the loss function and ‘adamax' as the optimizer. Model training was done over 100 epochs where maximum training accuracy of 99.5% was achieved. The training and validation, accuracy and loss curve graphs are displayed below.

**Fig 3.** Accuracy and Loss curves.



****

**MobileNet**

MobileNet is a computer vision model open-sourced by Google and designed for training classifiers. It uses depthwise convolutions to significantly reduce the number of parameters compared to other networks, resulting in a lightweight deep neural network. MobileNet is Tensorflow’s first mobile computer vision model.

It is a lightweight [deep neural networks](https://builtin.com/machine-learning/what-is-deep-learning).

A depthwise separable convolution is made from two operations:

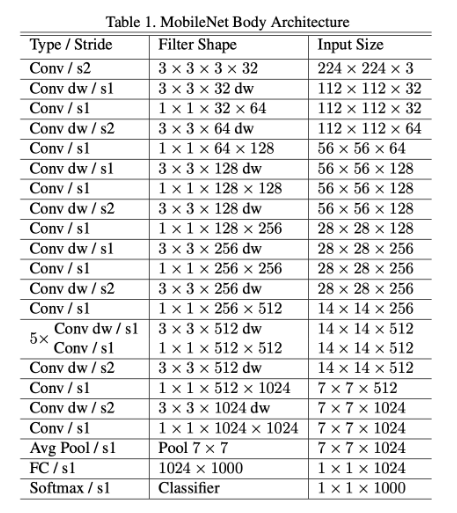
1. Point-wise convolution

2. Depth-wise convolution

The speed and power consumption of the network is proportional to the number of multiply-accumulates (MACs), which measures the number of fused multiplication and addition operations.

ARCHITECTURE:

The MobileNet model is based on depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution. For MobileNets the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs with the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size.



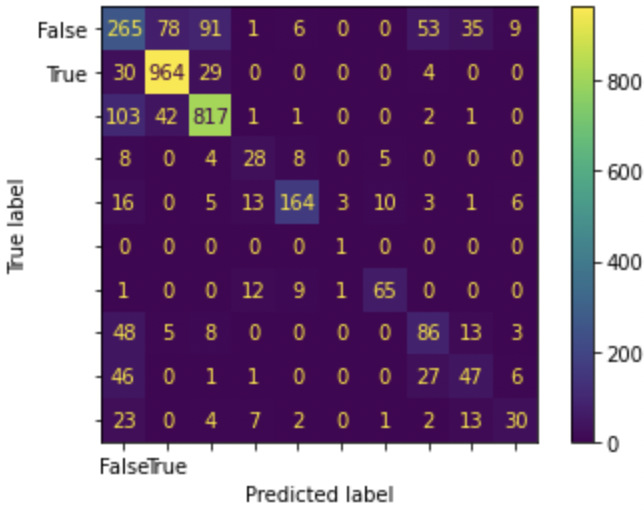
1. **Experimental Results:**

In this study, we implemented SqueezeNet and MobbileNet on AstroNN’s SDSS dataset for classifying Galaxies. The purpose of this study was to evaluate the performance of these models on this dataset and compare its accuracy with other state-of-the-art deep learning models.

**SqueezeNet:**

We evaluated the performance of SqueezeNet on the test set and achieved a training accuracy of 99.5% and an overall test accuracy of 80%. The confusion matrix and classification report for the classification results is shown below.

**Fig4.** Confusion Matrix for Squeezenet



**Table 3.** Classification Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| 0 | 0.61 | 0.63 | 0.62 | 536 |
| 1 | 0.90 | 0.91 | 0.91 | 1029 |
| 2 | 0.88 | 0.88 | 0.88 | 972 |
| 3 | 0.71 | 0.44 | 0.54 | 50 |
| 4 | 0.85 | 0.88 | 0.86 | 225 |
| 5 | 0.00 | 0.00 | 0.00 | 1 |
| 6 | 0.87 | 0.85 | 0.86 | 103 |
| 7 | 0.54 | 0.53 | 0.54 | 148 |
| 8 | 0.52 | 0.50 | 0.51 | 133 |
| 9 | 0.57 | 0.61 | 0.59 | 71 |
|  |  |  |  |  |
| Accuracy |  |  | 0.80 | 3268 |
| Macro Avg | 0.65 | 0.62 | 0.63 | 3268 |
| Weighted Avg | 0.80 | 0.80 | 0.80 | 3268 |

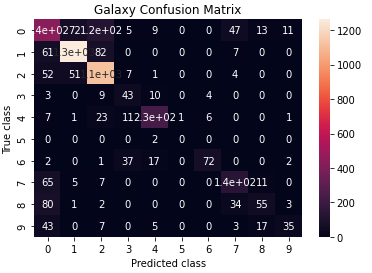
**MobileNet:**

We evaluated the performance of MobileNet on the test set and achieved for the test data(50 epochs ) and train data as follows :

Train data Accuracy : 96.79%

Test data Accuracy : 78%

The confusion matrix for the classification of galaxy using MobilNet is as follows:



**DISCUSSION**

While achieving a test accuracy of 82.4% with CNN architecture is certainly commendable, there is always room for improvement. One potential area of improvement is the use of Squeezenet architecture, which is known for its efficiency and low parameter count. However, the trade-off for such a lightweight architecture is often lower accuracy. Therefore, it is important to carefully balance the need for accuracy with the limitations of the architecture or the hardware on which the model will be deployed.

In the case of our research project, deploying the model on a website or mobile app as a lightweight model is a feasible solution. Although the accuracy may be lower, the benefits of making the model accessible to citizen scientists and increasing the number of people who can contribute to galaxy classification is significant. This approach is especially valuable for large-scale citizen science initiatives and community-driven projects. Overall, by using efficient architecture and deploying models on lightweight platforms, we can not only improve the accuracy of classification but also engage a broader audience in the scientific process.

**5.Conclusion and future scope:**

Deploying a galaxy morphology classification model on a web or mobile app can open up several exciting possibilities for future scope and user contributions. Here are some potential avenues to explore:

1. Increased User Participation: By making the classification model available on a user-friendly platform, such as a web or mobile app, more people will be able to contribute to the project. This could greatly increase the amount of data that can be classified, which in turn could lead to a more accurate and comprehensive model.
2. Real-Time Classification: Edge computing can be used to classify galaxy morphology in real-time. By deploying the model on mobile devices or on the edge of the network, users can contribute to the classification process even when they are offline. This can greatly increase the amount of data that can be classified, and it can also reduce the computational burden on the central server.
3. Improved Model Accuracy: With more data and more user input, the model can be trained to be more accurate over time. As users classify more galaxies and provide feedback on the accuracy of the model's predictions, the model can be fine-tuned to improve its performance.
4. Educational Opportunities: Deploying the model on a web or mobile app can provide an opportunity to educate the public about galaxies and their morphology. By providing a user-friendly interface and educational resources, users can learn about the science behind galaxy classification while contributing to the project.
5. New Research Directions: By making the classification model available to a wide range of users, new research questions may emerge. For example, researchers may be able to use the data to study the distribution of galaxy morphology in different regions of space or to investigate the relationship between galaxy morphology and other astronomical phenomena.

Conclusion

In conclusion, the use of efficient architectures such as Squeezenet can be a valuable tool for improving the accessibility and accuracy of classification models, especially in citizen science initiatives and community-driven projects. While there may be trade-offs in terms of accuracy, the benefits of engaging a broader audience in the scientific process and increasing the number of people who can contribute to galaxy classification are significant. By carefully balancing the need for accuracy with the limitations of the architecture or the hardware on which the model will be deployed, we can make significant strides in improving the efficiency and accessibility of classification models. Overall, the integration of lightweight models and efficient architectures has the potential to democratize the scientific process and bring us closer to a more comprehensive understanding of the universe.

**7. Gantt Chart**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tasks** | **Start** | **End** | **Dec 22** | | | | **Jan23** | | | | **Feb23** |
| **Research paper’s study on Galaxy detection using Deep learning** | **29/11/22** | **5/12/22** | **W1** | **W2** | **W3** | **W4** | **W5** | **W6** | **W7** | **W8** | **W9** |
|  |  |  |  |  |  |  |  |  |
| **Implementing yolo model for basic understanding of galaxy detection** | **6/12/22** | **20/12/22** |  |  |  |  |  |  |  |  |  |
| **Implement our own model architecture based on Neural network** | **21/12/22** | **27/12/22** |  |  |  |  |  |  |  |  |  |
| **Research on Squeezenet, Sparknet architecture and learnt about separable convolution** | **28/12/22** | **10/1/23** |  |  |  |  |  |  |  |  |  |
| **Worked on Squeezenet and mobilenet model for better accuracy** | **11/1/23** | **17/1/23** |  |  |  |  |  |  |  |  |  |
| **Final report preparation** | **18/1/23** | **6/2/23** |  |  |  |  |  |  |  |  |  |
|  | | | | | |  | | | | | |