**Anthology Made Easy**

**(An Image Classification Algorithm Development and Comparison)**

***Abstract* -We have designed a Web Application where users can select an image and retrieve the details of the flower such as different kinds of scientific names, Description, and Interesting facts. In this paper, we have compared different Convolution Neural Network architectures such as LeNet-5, Alex Net with the model designed by us for classification purposes. We shall discuss the results and observations noted down when comparing these architectures and also on the step-by-step process on how the model was modified to eventually increase the accuracy and reduce the loss**.

**Keywords** - Image classification, CNN, Anthology, flower dataset, CNN, Comparison of Models

1. **Introduction**

There are more than 4,00,000 plant species available in the world. In the world of Anthology, identifying and discerning the vast number of available flowers is not easy. A simplified approach to this problem would be to make use of technology to classify and recognize images. Image Classification has come a long way in the last 2 decades from being able to identify simple faces to be able to recognize ancient languages and translate them. The initial image classification algorithms were inaccurate, huge and resource consuming. As more bright minds started working on this, more efficient algorithms were developed. There was a breakthrough in 2012 when CNNs were used in an ImageNet competition wherein this algorithm gave a record low error rate of 16.4% [1]. In the following years, the error rates in image classification fell to single digit percentages and ever since 2012, have always been convolutional neural networks. We are attempting to build a model that can accurately recognize a flower based on the image we feed to the model. We are accomplishing this through building a GUI for the user to interact and get a result in a readable format.

1. **Literature Review**

Previously work has been done using segmented technique in [2] to classify the image of flowers. Google Net and Alex Net CNN architectures were compared. In the comparison Google Net performed better than Alex Net. Stochastic gradient descent algorithm was used to train Google Net. The dataset contained black background with an image of a flower. The Stochastic pooling technique has been carried out in [3] using CNN model. The model used four convolutional layers, each layer with different filtering size. Training and Validation loss was observed less with that CNN architecture.

A completely new and different perspective of image classification has been used in [4] using Gray level co-occurrence matrix (GLCM) and discrete wavelet transform (DWT). The textures features are extracted using GLCM and DWT techniques which are used for training the model using Multilayer Perceptron Neural Network. In the classification phase, the Artificial Neural Network classifier is used with the backpropagation algorithm. This model predominantly works with grayscale images in the training phase and hence has a limited use. But the model has a better performance as the backpropagation algorithm is being used.

Models such as Alex Net and LeNet-5 have a definite architecture [5]&[6] which was successful, but our quest for constant improvement made us create our own architecture which best suits our scenario. These architectures gave us insight on activation functions such as ReLU and Tanh, pooling techniques like average pooling and max pooling with different convolutional layers.

We have opted for the experimental approach in our paper and train model in LeNet 5, Alex Net and models trained by us. We have augmented datasets using augmented technique and gave the augmented data sets as input in our CNNs to improve the performance of our model. All the 3 models have different architectures. We have presented our findings on all the 3 models and performance of the models.

1. **Data**

The data we have used is a collection of 1360 images split into 17 categories each, i.e., the “17 Category Flower Data Set” [1] provided by the University of Oxford. They are the flowers most commonly found in the United Kingdom. The dataset contains 17 categories of flowers each having 80 images of that category. The images are identical yet different in terms of their size, angle of capturing, lighting etc. The speciality of this dataset is it has labels associated with it.

The list of flowers available are Daffodil, Snowdrop, Lily, Valley, Bluebell, Crocus, Iris, Tigerlily, Tulips, Fritillary, Sunflower, Daisy, Colts, Foot, Dandelion, Cowslip, Buttercup.

1. **Methodology and Results**

Now that we have discussed the dataset and the different methods used in Image Classification, let us move on to our model and the different methods used in our approach. Since we have used the augmentation technique on our dataset, we have expanded our dataset from 80 images to approximately 395 images per flower. In our model, while reading each image, we do some operations on the images to standardize our image data while feeding it into the model. We first resize the image to a specific size and then convert it into a Numpy array for easier and faster operations on our dataset.

After the dataset was ready, we divided it into training, testing and validation data. We considered 80% of our dataset for training, remaining 20% for testing. We have used a combination of Convolution 2D layers and Max Pooling layers in our CNN mode since that is the standard approach while dealing with image classification techniques. Along with the convolution layers, is passed on to two fully connected Dense layers. The number of neurons in the output layer should be equal to the number of categories of data we are trying to classify. We have used MaxPooling instead of AveragePooling because former is believed to detect sharper edges more effectively and the latter is supposed to be better for tackling smoother edges.

Initially, the model was trained using un-augmented data. i.e., 80 images per each flower. We used a model that consisted of  5 layers (combination of 5 Conv2D and 5 MaxPooling layers) which returned a validation accuracy which was too less and a validation error which was too high, which is not ideal . We then reduced the number of layers to 4, but could see no visible improvement in the accuracy or the error. This is when we decided to augment our data, and to continue using 4 layers for our future models.

Using this data, did improve the performance of our model, but with significant validation error. This affected the prediction accuracy of the flowers. Hence to improve our model, we started changing parameters of the layers, example, changing the number of filters in a Conv2D layer or changing the stride size for a MaxPooling2D layer. After reading the images using opencv, we resize the images to 50\*50.

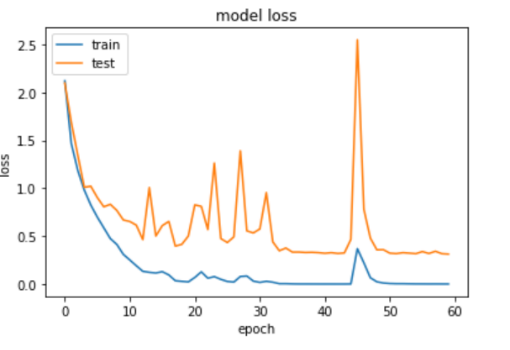
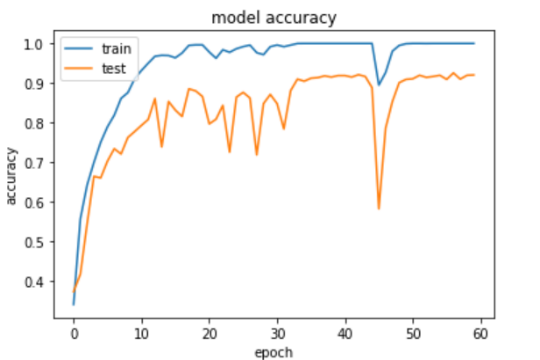
8 layers having Conv2D and MaxPooling2D layers. We have used different filters and filter sizes for each Conv2D layer. We also varied the stride sizes for MaxPooling2D layers for different models. We started with 64,128,128,256 filters for the 4 Conv2D layers respectively and a 3\*3 stride size for a MaxPooling2D layers and 1\*1 stride size for the remaining MaxPooling2D layers. The accuracy and loss of this model was not up to the mark and hence we decided to change a few parameters for our next model. For our next model, we changed the filter to 256,256 for the third and fourth layer respectively, but without any significant improvement in the model’s accuracy or error. To make our model perform better, we changed the filter to 64, 128 for the second and third layer respectively without any change in the MaxPooling2D layers. This model has a significant increase in the accuracy and decrease in error. To further increase our model’s performance, we changed the filter to 128,64 for the first and third Conv2D layer. Though our results gradually improved, to make the model more accurate with less error, these models are trained using grayscale images instead of the RGB images.

Once we have explored our model based on the RGB images, we then sought out to test the robustness of our model on grayscale images. We did not use a dataset of grayscale images, but rather converted the images into grayscale while reading each image. Reading the images in grayscale improved our system performance since the model now has to deal with only 1 channel.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Image Type** | **Changes Done** | **Accuracy** |
| Model 1 | RGB | Started with 64,128,128,256 filters for the 4 Conv2D layers | 70.2% |
| Grayscale | 62.1% |
| Model 2 | RGB | Change in third and fourth Conv2D layer to 256,256 | 72.1% |
| Grayscale | 63.9% |
| Model 3 | RGB | Change in second and third Conv2D layer to 256,256 | 73.9% |
| Grayscale | 65.7% |
| Model 3 | RGB | Change in first and third Conv2D layer to 256,256 | 81.4% |
| Grayscale | 64.2% |

**Table 1: Experimentation of models**

As per the above table, we can observe that all the models fared well on coloured images rather than grayscale images. This is because the dataset is primarily of flowers and sometimes flowers may look the same, based on edges and shapes, but may differ in colours, so considering the RGB channels will help the model in accurately identifying the flower. H. M. Bui et al. [1], has suggested that, using grayscale images will help achieve higher classification accuracy and reduce the computational cost, but this does not seem to be right in our case. Hence, we started exploring our model a little more, now with just RGB images and reducing the number of layers used for the model. A graphical representation of such a model can be observed in the below graphs, with almost 90% accuracy and around 30% error.



**Figure 1: Model Accuracy and Loss**

While designing our model, initially we have been using 60 epochs. Once we plotted our Accuracy and Loss values, we noticed that the accuracy and loss values were pretty much stable after 35 epochs. In both the values, we could see a sharp spike around the 45th epoch and then the values bounce back. Post this, we decided to run the model up to 35 epochs to increase the efficiency and reduce the computational resources.

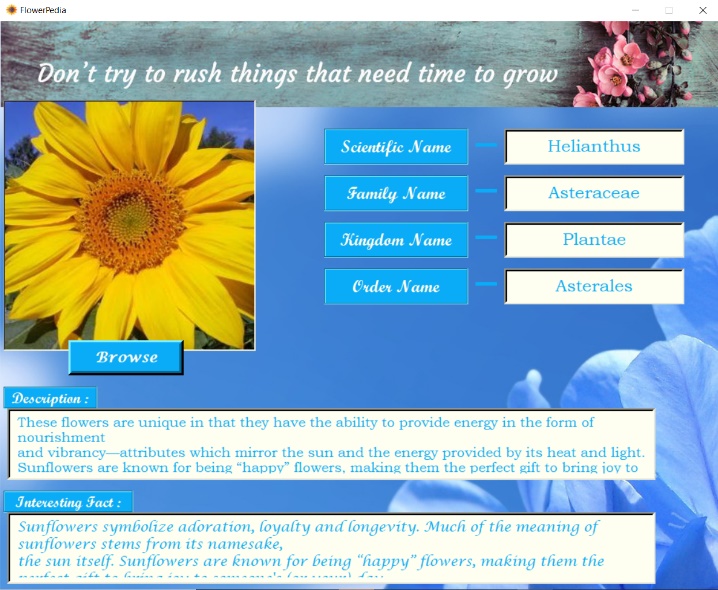
An analysis was done with the existing standard models for our dataset to see the performance between various  models.

|  |  |  |
| --- | --- | --- |
|  | **Image Type** | **Accuracy** |
| LeNet-5 | RGB | 52.64% |
| Gray Scale | 38.96% |
| Alex Net | RGB | 4.69% |
| Gray Scale | 5.88% |
| Our Model | RGB | 90.3% |
| Gray Scale | 72% |

**Table 2: Comparison of Models**

If we notice the above table, we could see that the models with lesser number of layers like LeNet-5, which has only five layers, have a low accuracy value. For LeNet-5, the accuracy for colour images is 52.64%, whereas the accuracy for grayscale images is 38.96%. For Alex Net, we see a stark difference in the accuracy percentage, and we believe it is because originally, Alex Net was trained on millions on images. Ours is a very limited data set with around 390 images per flower so this might be the reason why such an efficient algorithm fails, because of limited data. Our final model is much more efficient in recognising the images. This was achieved by gradually tweaking and adding/removing layers.

The final GUI looks as per the below picture. We are using the Tkinter library of Python to build the interface. We use opencv to read the image from our local and pass the image to our model after resizing the image.



**Figure 2: GUI**

Once this image is passed to our model, it predicts the flower and returns the name for the same. This name is then passed to a google search to retrieve various details of the flower such as Scientific Name, Family Name, Kingdom Name, Order Name, Description, and an Interesting Fact about the flower. We use Beautiful Soup library of Python for web scraping.

Since all google web pages follow a standard format of tags and structure, we can determine where our required data is present once we study the tag structures. Once this information is available, we use Beautiful Soup to extract the text. This text is then passed to the GUI to be displayed in appropriate labels.

1. **Limitations**

The dataset in our possession was very limited with each category having only 80 images. Initially, this led to our model having a low accuracy. There were other datasets available, but they were unstructured, with uneven number of images per category and missing labels and hence of little value. Because of our limited data set, we had to use the Data Sampling process to increase our data size.

Depending on the dataset, we must design our model carefully. The VGG-16 model could not be run on our systems as it had too many parameters. This led to over consumption of the resources and threw an Out of Memory error.

1. **Recommendations**

The first step to do when designing the model is to have a complete dataset at hand. It is always advantageous to have a large dataset with properly assigned labels so that our model has a larger dataset to train on. Once we collect the required data, we need to build a robust and a strong algorithm to train our models. Often, our models will have multiple layers, and more the layers, stronger the system. So, in order to create a robust and a stronger model, we need to have a system with powerful specifications to handle the millions of parameters for certain topology of models.

There are multiple IDEs and web-based scripting platforms like Jupyter notebook, Spyder or Pycharm to write the Python code and to share among peers. But these tools are limited by your system specification and cannot handle more data. Google provides a platform called Google Colab, where the Machine Learning enthusiasts can run their model without any hindrance of system hardware limitations. One of the main features of this Colab is that the users can simultaneously work on the same notebook for version control and it is very fast when run on GPU mode.

We would like to suggest you investigate Teachable Machine; an automated model generator provided by Google. It is a very powerful tool to generate Neural network architectures for image classification based on the input images you provide.

**References**

[1] Bui et al., "Using grayscale images for object recognition with convolutional-recursive neural network," 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), Ha Long, 2016, pp. 321-325.

[2] Gurnani et al., “Flower Categorization using Deep Convolutional Neural Networks”, 2017, arXiv:1708.03763

[3] Prasad et al., “A efficient classification of flower images with convolutional neural networks, International Journal of Engineering and Technology, 7 (1.1) (2018) 384-391

[4] Archana L. ,Lakesar, “A Review on Flower Classification Using Neural Network Classifier”, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

[5] Yann LeCun et al., “Gradient Based Learning Applied to Document Recognition”, PROC OF THE IEEE NOVEMBER 1998

[6] Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks”, [Advances in Neural Information Processing Systems 25](http://papers.nips.cc/book/advances-in-neural-information-processing-systems-25-2012), 2012