AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Soft Computing and Machine Intelligence Journal Vol (X), Issue (X), 20XX

Flower recognition using VGG-13 with 90%+ accuracy

Latif, Nayeem Bin Abdul, ID: 17-35801-3 [a]

(a) Gazipur, Dhaka

* Nayeemlatif@yahoo.com

Abstract: This particular research used a database of over 4000 flower pictures to classify five different types of flowers. The purpose of this project was to improve accuracy over a prior Kaggle experiment. The project was a success, with over 90% training accuracy and 81% validation accuracy. Before the recommended model, VGG-13, was chosen, a series of tests were done. It's a much simpler form of VGG-16,19 that was chosen to avoid overfitting problems. The paper demonstrates the VGG model's development and implementation, as well as how the datasets were trained. When an image travels through the VGG model, the progress of the layers is seen

Keywords: Flower Recognition, Accuracy, CNN, VGG-13 Model

1. Introduction

In today's modern world, advancements in all fields of study, including technological advancements, occur at a dizzying rate. Among them computer vision or technologies that work with visual imagery is becoming ever more popular such as facial recognition, disease prediction and recognition of any visual things. Visual information usually forms images or imagery, the development of imagery data growth can help people recognize an object so that it can present Visual information [1]. Flower recognition or classification is a very common subject in the field of computer vision. Despite its popularity it's a quite difficult subject because of the huge types of flowers in the world. Most flower are unique to their looks size and color so working with few types can be tough. After much research this subject was selected because of how unique and difficult the problem is. This study deals with five different type of flower and their dataset over four thousand visual imageries. The goal is to classify these flowers with more accuracy than prior projects in this field. At first a prior project is chosen to be the standard for the training accuracy and validation accuracy. Then to exceed the accuracy, a new model has been proposed. The proposed model is VGG-13 while the standard model is classic convolution neural networks with bit modification. So, the report will contain the background information about the CNN model and the standard project that has been chosen. It will also contain the new proposed model and their result. Finally, a comparative discussion on how the proposed model obtain higher accuracy. The link for the Kaggle project that has been chosen for standard is given: Link: https://www.kaggle.com/rajmehra03/flower-recognition-cnn-keras/notebook

2. Literature Review

Flowers come in such a wide variety of colors and shapes that identifying them can be challenging. Knowing how to identify a flower without the guidance of a botanist would be extremely beneficial to sectors such as medicines and cosmetics. One of the most defining characteristics of a flower is its color. Avishiktha Lodh and Ranjan Parekh wrote a work that took this into mind (2017) [2]. Various algorithms have been developed to address the problem of flower classification up to this point. Several

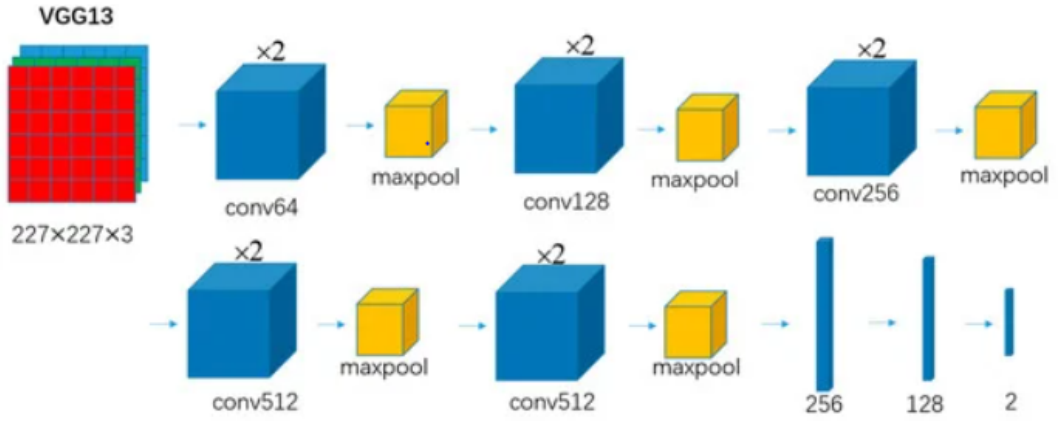
classic image categorization algorithms based on handcrafted characteristics have been proposed. Guru et al. [3] suggested a system for categorizing flower photos based on textural features. They tested their procedure on 35 different varieties of flowers to see how effective it was. With their system, they could reach a maximum result of 79 percent accuracy. There are lots of studies on this field some of them exceed very high accuracy with the help of deep CNN and other newly discovered model. Those aren't included since this projects aim is to get higher accuracy than its standard model. The standard model whose title is 'Flower Recognition CNN Keras' is taken from Kaggle [4]. It's a three years old project done on flower dataset which contain 5 different flower type with over 4300 images. The five types are daisy, rose, tulip, dandelion and sunflowers each contains average of 865 images. At first, they prepared the dataset then they used 'ImageDataGenerator' for preprocessing the data. The model for the training that they used is a normal ConvNet model with a little modification. Their model has six layers, four of which are convolutional layers, as well as input and output layers. Between the convo layers is a maxpooling layer, and following the convo layer is a flatten layer. They employed the 'relu' activation for conv layers and the 'softmax' activation for output [4]. They trained the model with optimizer 'Adam' for 50 epoch and obtained an accuracy of 86% and validation accuracy of 78%. This is an outstanding result for a standard ConvNet model. This project will be aiming to get higher accuracy then this by using different training model. Many trials and errors were conducted in order to achieve this, and the vgg-13 model was chosen after much investigation.

3. Proposed Model

Since the project is about image classification, there were a variety of CNN models to pick from, including ResNet, DenseNet, AlexNet, VGG, and others. These models may be augmented and modified to do certain tasks. The standard model, for example, was based on a traditional ConvNet model that had been prepossessed and modified. After much trying, for this particular case a modified VGG model was chosen. Specifically, VGG-13 model with slight modifications.

4. VGG-13

Simonyan and Zisserman introduced the Visual Geometry Group (VGG) Network design. VGG Net's design enables it to execute with excellent precision. The architecture created by Visual Geometry Group has 6 types of architecture. A layer of repeated convolution and pooling is present in the design. VGG-13 Net has 13 layers, 10 convolution layers and 3 fully connected layers, while VGG-16 Net has just 13 convolution layers and 3 fully connected layers[6]. The depth of the network is a crucial aspect for obtaining high performance, according to a deep-structure VGG-Net. A figure of VGG-13 model architecture is given below:



5. Implementation

Due to technological issues, the implication process was completed with the assistance of Google Colab. Because the flower dataset had to be transferred to Google Drive, the procedure took a little longer. They were resized to 100 by 100 after importing the dataset to ensure that all photographs were the same size. They were then transformed to a np array. Each flower category was divided into three folders: train, test, and validation. After that, the VGG-13 model was used. Some picture preprocessing was done before training the model. SGD optimizer was used to train the model for 100 epochs until the required result was obtained.

6. Result and Discussion

6.0.1. Results

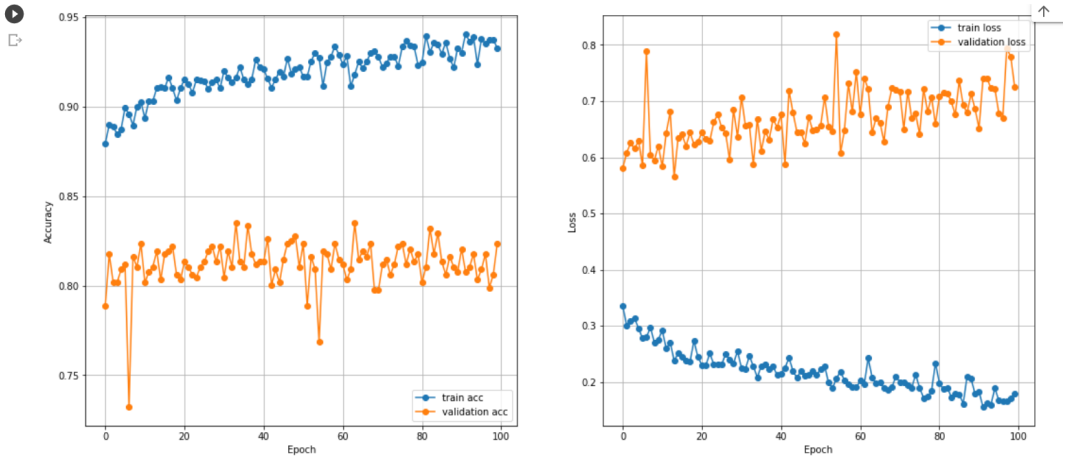
The purpose of this project is to achieve a greater level of accuracy than a previously chosen project using the same dataset. Because of that, the results of training and validation accuracy are shown first.

```
Epoch 48/50
19/19 [==============================] - 14s 747ms/step - loss: 0.3701 - acc: 0.8616 - v
al_loss: 0.6822 - val_acc: 0.7804
Epoch 49/50
19/19 [==============================] - 14s 740ms/step - loss: 0.3353 - acc: 0.8736 - v
al_loss: 0.5710 - val_acc: 0.8135
Epoch 50/50
19/19 [==============================] - 14s 745ms/step - loss: 0.3410 - acc: 0.8676 - v
al_loss: 0.6586 - val_acc: 0.7898
```

As shown in the figure, after 50 epochs of training, the greatest accuracy attained was 86 percent, while validation accuracy was 78%. As a result, if the suggested model improves in accuracy and val accuracy, the project aim will be achieved. The VGG-13 model training results are shown below.

```
+ Code  + Text
44/44 [==============================] - 10s 228ms/step - loss: 0.1826 - acc: 0.9301 - val_loss: 0.6517 - val_acc: 0.8205
[13] Epoch 92/100
44/44 [==============================] - 10s 228ms/step - loss: 0.1557 - acc: 0.9406 - val_loss: 0.7400 - val_acc: 0.8075
Epoch 93/100
44/44 [==============================] - 10s 228ms/step - loss: 0.1630 - acc: 0.9363 - val_loss: 0.7410 - val_acc: 0.8104
Epoch 94/100
44/44 [==============================] - 10s 228ms/step - loss: 0.1595 - acc: 0.9388 - val_loss: 0.7229 - val_acc: 0.8177
Epoch 95/100
44/44 [==============================] - 10s 227ms/step - loss: 0.1891 - acc: 0.9236 - val_loss: 0.7221 - val_acc: 0.8032
Epoch 96/100
44/44 [==============================] - 10s 227ms/step - loss: 0.1671 - acc: 0.9377 - val_loss: 0.6785 - val_acc: 0.8090
Epoch 97/100
44/44 [==============================] - 10s 227ms/step - loss: 0.1658 - acc: 0.9352 - val_loss: 0.6696 - val_acc: 0.8177
Epoch 98/100
44/44 [==============================] - 10s 229ms/step - loss: 0.1655 - acc: 0.9374 - val_loss: 0.7937 - val_acc: 0.7988
Epoch 99/100
44/44 [==============================] - 10s 228ms/step - loss: 0.1711 - acc: 0.9374 - val_loss: 0.7795 - val_acc: 0.8061
Epoch 100/100
44/44 [==============================] - 10s 227ms/step - loss: 0.1800 - acc: 0.9327 - val_loss: 0.7251 - val_acc: 0.8234
```

The project version of model training is depicted in this figure. As can be observed, the maximum average accuracy of 93 percent was achieved after 100 epochs. Validation accuracy, on the other hand, was an average of 81 percent. To achieve the best and most consistent results, the model was trained for 100 epochs. It is noted that training this model again would change the result a bit. The graph of the training model is shown below. It contains accuracy, validation accuracy curve and loss percentage and validation loss percentage.

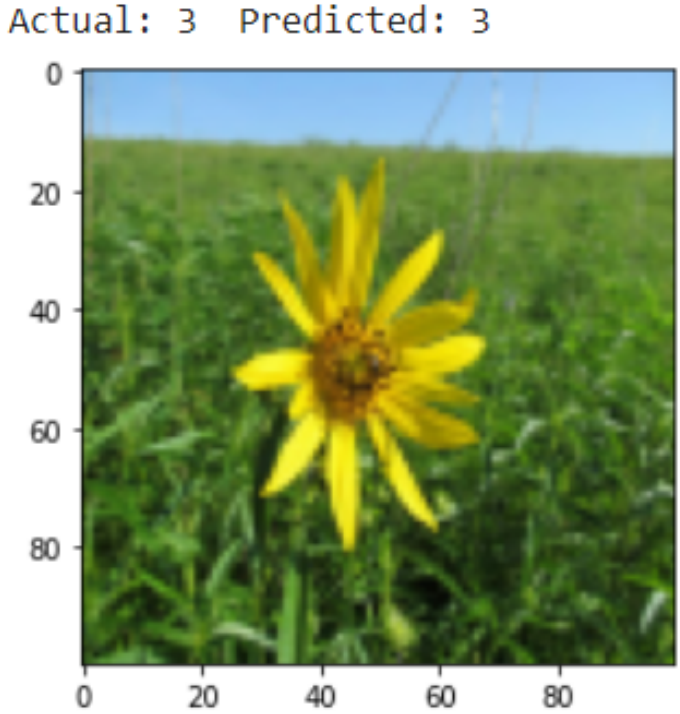


We can observe from this graph that the accuracy increased from 86 to 94 percent at one time. While validation accuracy remained essentially unchanged, which is disappointing because the higher the accuracy, the better. Despite the lack of val precision, the results were quite consistent, reaching 82 percent. The model had already been trained, which explains why accuracy is at 86 percent. After three training sessions, the last one yields the best benefits. The model has been trained for over 150 epochs in total. The percentage of loss is bad because it can be observed a little more clearly from the val accuracy. However, because the outcome was consistency, it was still far superior than any other model that has been trained on the dataset. For the model compiler the optimizer 'SGD' was used as it is better doe VGG model. The images were preprocessed before training with help of 'ImageDataGenerator'. Prepressing was crucial for achieving the greatest results. A few other processing versions had been attempted, but some tweaking was needed to get the job done. The figure of compile method and image preprocess method is shown.

```
[ ] model.compile(loss='categorical_crossentropy',optimizer='sgd',metrics=['acc'])
```

```
[ ] datagen = ImageDataGenerator(
            featurewise_center=False,
            samplewise_center=True,
            featurewise_std_normalization=False,
            samplewise_std_normalization=False,
            zca_whitening=False,
            #rotation_range=10,
            #zoom_range = 0.1,
           # width_shift_range=0.2,
           # height_shift_range=0.2,
            horizontal_flip=True,
            vertical_flip=True)
    datagen.fit(X_train)
```

After training the model a code has been ran for predicting a single image to check if the model's accuracy is good. And it can be seen that predicting image and the actual image are the same. In this case the model predicted a sunflower and the actual image is sunflower as well.



6.0.2. **Unsuccessful Experiment**

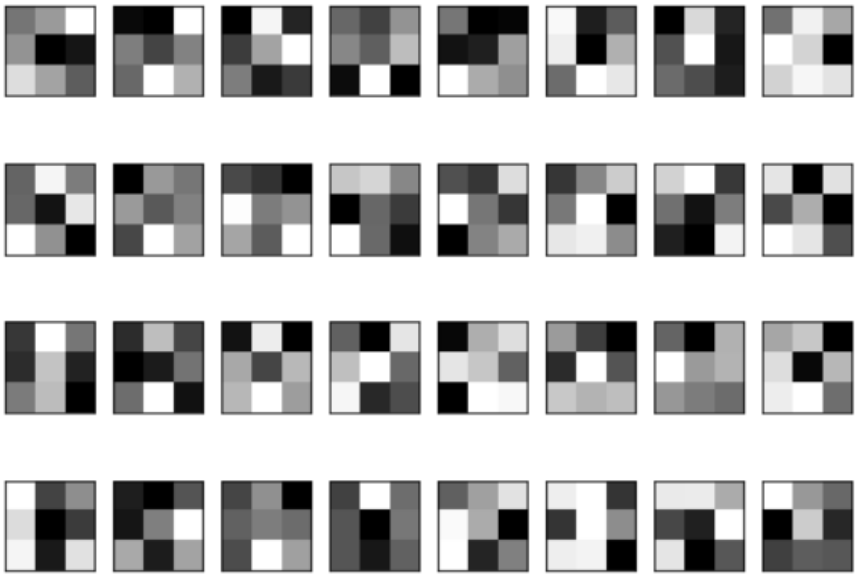
This project's goal was achieved through a series of unsuccessful experiments. The dataset was chosen for its simplicity, as it is expected to function with a variety of CNN models. However, such assumption was incorrect. Before vgg-13, a few alternative models, such as AlexNet and DenseNet, were used. Even after 50 epoch or so, the accuracy of both of these models was fairly poor. Following considerable investigation, it has been determined that the VGG model is the best for this type of dataset. As a result, VGG-16 was used first since it is the most popular. An example of a figure from my experiment[7].

```
Epoch 199/200
46/46 [==============================] - 17s 378ms/step - loss: 0.0677 - accuracy: 0.9781 - val_loss: 2.3618 - val_accuracy:
0.6396
Epoch 200/200
46/46 [==============================] - 18s 395ms/step - loss: 0.0485 - accuracy: 0.9809 - val_loss: 3.0115 - val_accuracy:
0.5799
```

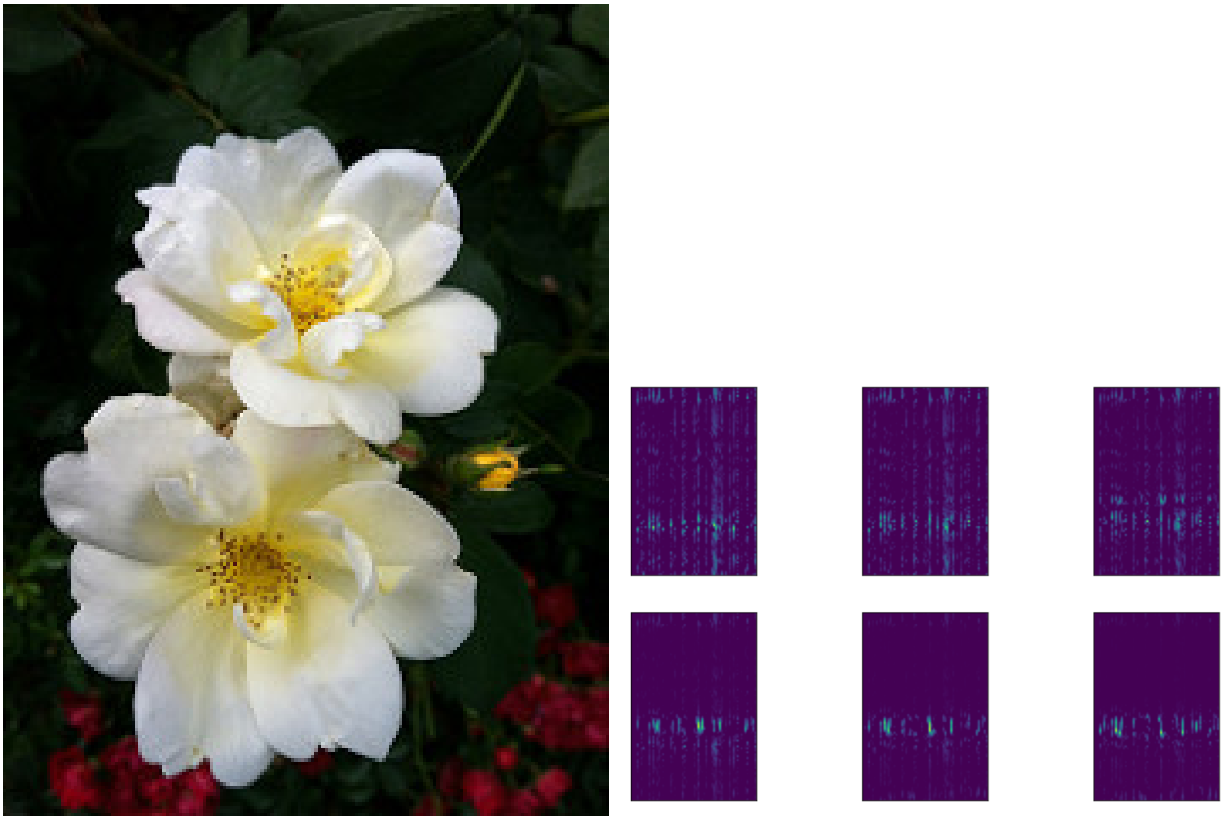
The validation accuracy was less than 60% even after 200 epochs, as seen in the figure. Even if the training accuracy is quite high in comparison to other models, if the val accuracy is low, the model is ineffective. The test accuracy and the training accuracy are both validated by Val accuracy. The model VGG-19 was also tested, although the findings were quite similar. Finally, vgg-13 comes in helpful. To arrive at the final outcome, three distinct optimizers were used. After extensive investigation, the optimizer 'SGD' was shown to be the most effective.

6.0.3. **Layer Visualization**

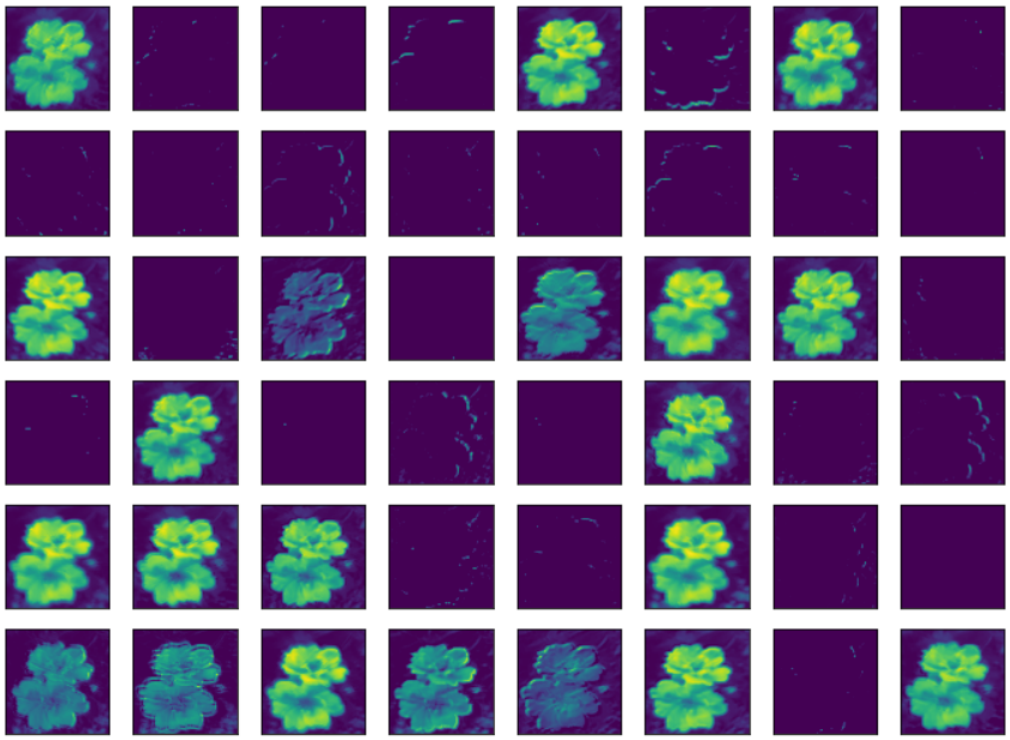
First the layers of all the filters have been showed. As seen from the figure 3*3 filters ere used in the convolution layers.



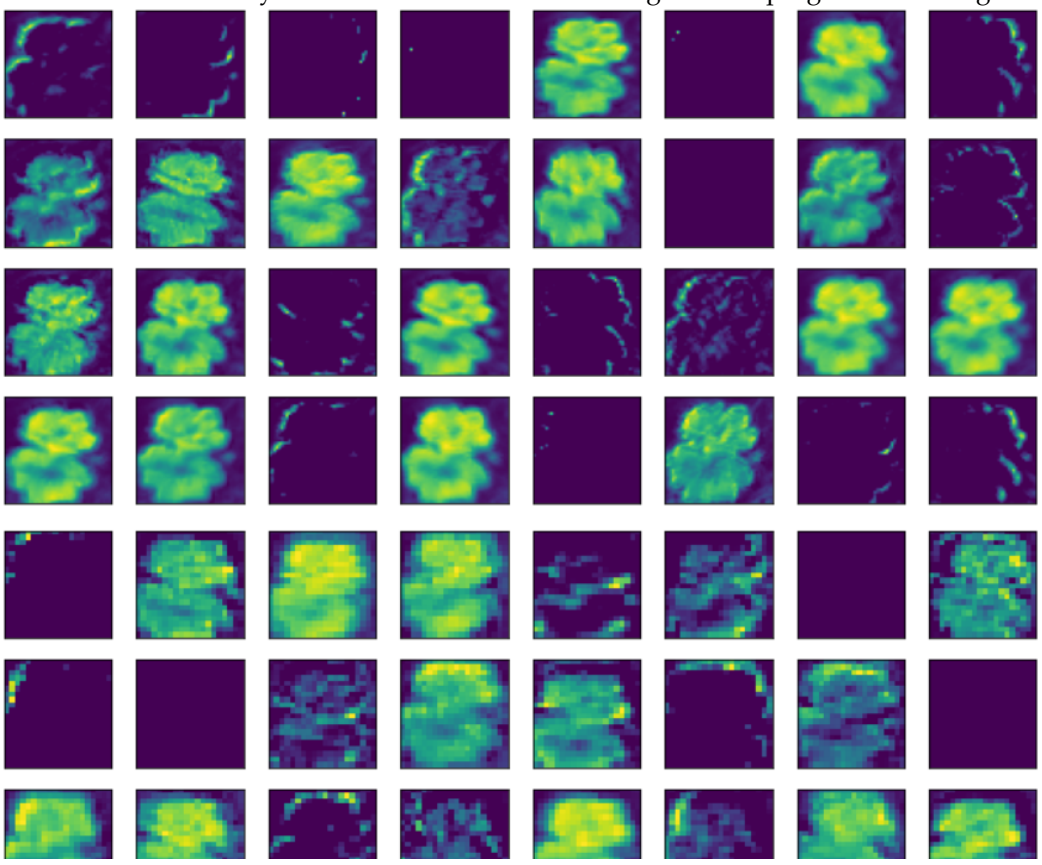
This part shows when the image passes through the model and layers, and how they change with each layer over time. It's the representation of a single image progressing through the model and multiple layers. Initially, a specific image from the rose genre was picked. The zoomed version of the first convolution layer is then displayed. Following that, a couple intermediate levels and the final convo layer are displayed.



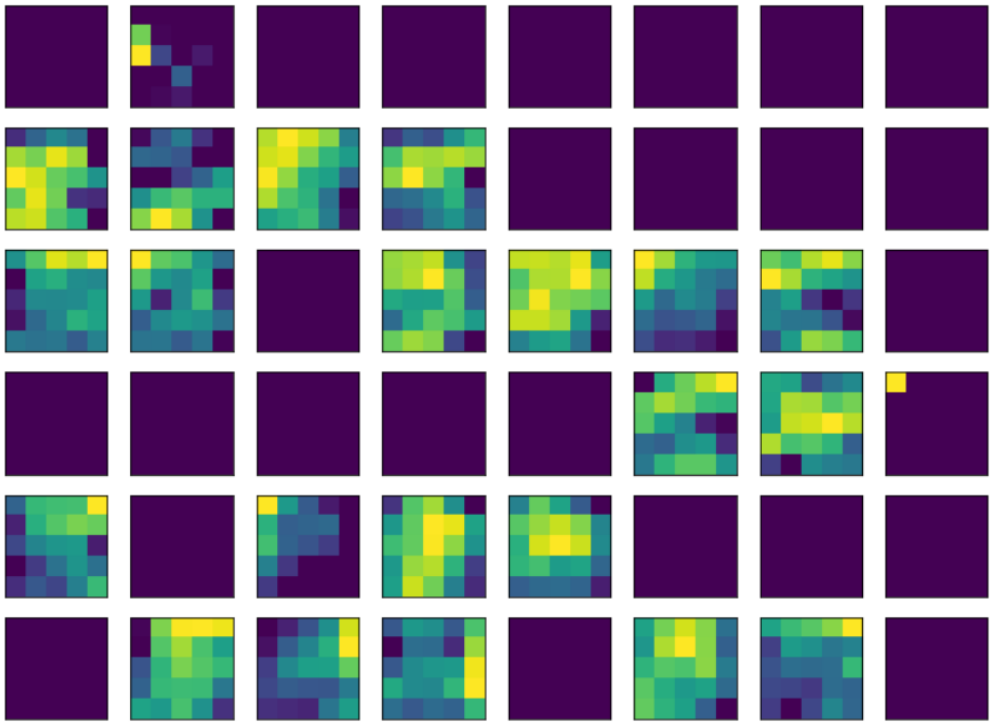
The first convolution layer visualization for the rose image and its progression through the layers.



The 5th convolution layer visualization for the rose image and its progression through the layers.

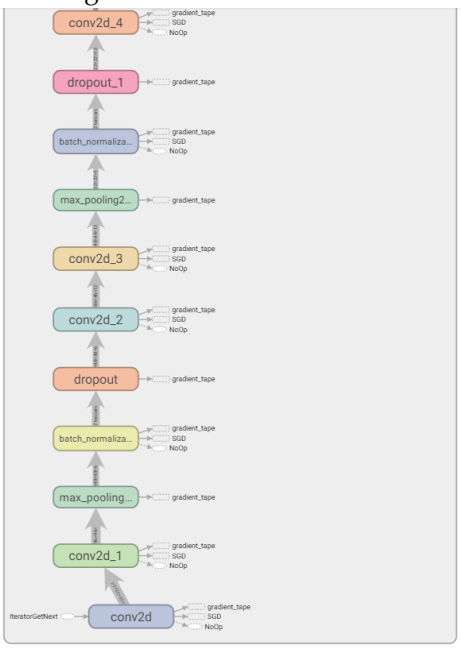


The Final convolution layer visualization for the rose image and its progression through the layers.



Because all of the layers would take up too much room, just three levels of the imaging process have been exhibited. The procedure reveals that the image was initially pretty distinct, but as it progresses through layers, it becomes increasingly blurry and unintelligible. The model dissected the image using filters to reveal its basic structure and form. After hundreds of pictures have been trained, the model can accurately detect and predict.

Tensorboard toolkit was used which is a tool for visualizing the model flow chart and its training process. Batch normalization, epoch number, accuracy and many flow chart for the model can be visualized through this toolkit. A flow chart for the VGG-13 model has been shown:



7. Arguments

The suggested model VGG-133 outperforms the typical CNN model, as evidenced by the model's results. From the result section it can be seen that both the training accuracy and the validation accuracy is doing better in the proposed model. The model VGG-13 performs better than its popular counterparts such as VGG-16,19, and this is due to the model's complexity. Because the VGG-13 has a lower conv layer than the VGG-16,19 it performs better in some circumstances. The VGG-16 and its precision, which VGG-13 sorely lacks, are also displayed in the results section. However, it should be emphasized that with additional augmentation and adjustment, the outcome might radically alter. Because the aim was to outperform the Kaggle project, the effort was a huge success.

8. Conclusion

Floral recognition using the considerably simpler VGG model may be useful for botanists and herbalists seeking for flower plants in the field. This study utilized a dataset of over 4000 flower photos to categorize 5 distinct types of flowers. The project was carried out with the goal of improving accuracy over a previous Kaggle project. With over 90% training accuracy and 81 percent validation accuracy, the project was a success. A number of tests were conducted before the suggested model, VGG-13, was chosen. It's a considerably simpler variant of VGG-16,19 that was chosen to reduce the problem of overfitting. When an image passes through the VGG model, it is visualized to demonstrate the layers' progress. Even if the project's aim has been met, there are other enhancements that may be made. A large-scale effort with a lot larger dataset would undoubtedly be both more difficult and useful.

9. Reference

1. Amir, S. "Hand posture classification with convolutional neural networks on VGG-19 net Architecture." IOP Conference Series: Earth and Environmental Science. Vol. 575. No. 1. IOP Publishing, 2020.
2. Lodh, Avishikta, and Ranjan Parekh. "Flower recognition system based on color and GIST features." 2017 Devices for Integrated Circuit (DevIC). IEEE, 2017.
3. Guru, D. S., YH Sharath Kumar, and S. Manjunath. "Textural features in flower classification." Mathematical and Computer Modelling 54.3-4 (2011): 1030-1036.
4. https://www.kaggle.com/rajmehra03/flower-recognition-cnn-keras.
5. Alipour, Neda, et al. "Flower Image Classification Using Deep Convolutional Neural Network." 2021 7th International Conference on Web Research (ICWR). IEEE, 2021.
6. https://neurohive.io/en/popular-networks/vgg16/
7. https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/