Flower Recognition with CNN and Transfer Learning

By

Simon Meng

INTRODUCTION

It's not a surprise that image recognition and classification has been part of our lives. In Amazon, we can take a picture of the object and it can tell us the price of the object, even the shopping page. In Google, one can input an image and the search engine can output thousands of threads related to it. Moreover, there are mobile apps in which users can take a photo (animal, tree, flower) and receive information about it.

In this project, we are aiming to accomplish a simple simulation of such apps in python, and build a simple UI in Jupyter Notebook. The problem statement, model selection, evaluation and summary will be explained in detail as followed.

PROJECT PROBLEM STATEMENT

Using machine learning, how can we identify flower species through images so that we can know what species it is on the fly.

BACKGROUND AND VALUE ADD

Before we had image recognition abilities, when we came across a flower, we need to memorize or snap a photo for it and describe it on Google, hoping someone else asked about the same flower before.

Now, this machine learning algorithm, with a proper UI design, would come in handy when it comes to field studies, hiking, daily lives, research and so on. It's fascinating to learn about the world around us, before it slips through our minds. These pieces of knowledge could be valuable assets to learn about the species, their habitats, where they originally came from.

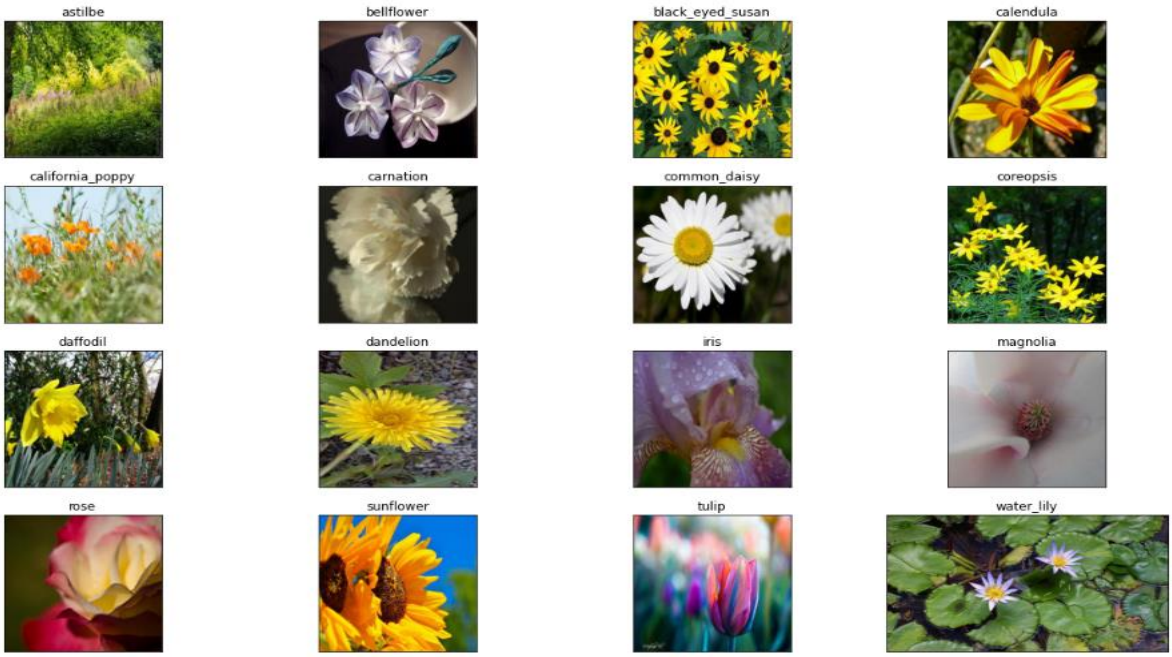
Furthermore, this model has the potential to be applied data in other fields such as fashion style data, which could be useful in e-commers area. Object detection, sometimes closely related to security, is also another area that could benefit from this algorithm. But most importantly, interest development and education for the general public would be the main purpose of this project.

DATA SOURCE

The data for this project was acquired publicly on Kaggle through the link https://www.kaggle.com/datasets/l3llff/flowers.

Our original data includes 15740 images of 16 flower species, each in a separate folder. Across all the species, iris has a max number of images of 1054, and astilbe a min of 737. Most images came in default 256 * 256, but some are in other sizes.

A sample of our flowers:

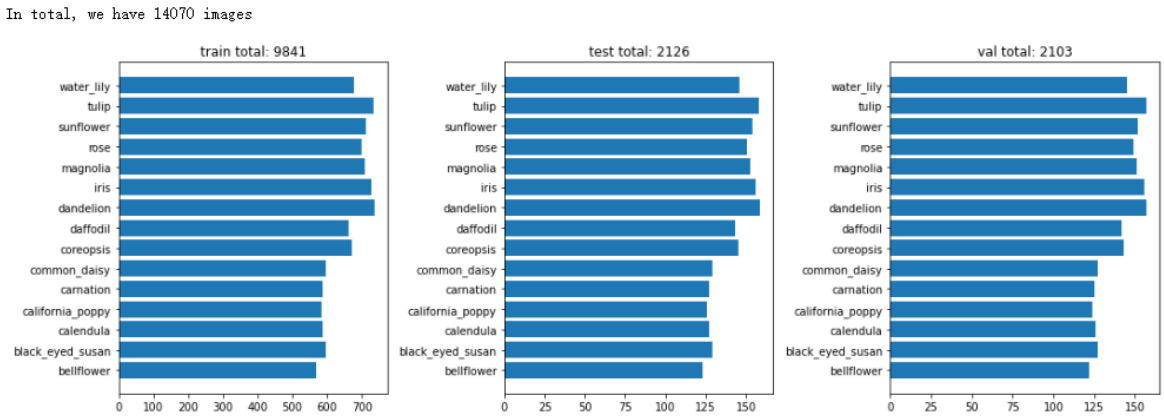


DATA PREPROCESSING

Our original data is a collection of daily live images (imagine what a typical photo you would take). Thus, some necessary clean up is needed to ensure data quality and integrity.

1. Removed irrelevant images
   Our model can be more focused without irrelevant objects, such as insects, leaves, people, houses, etc. Images with such objects are removed. Moreover, arts are removed.
2. Removed disoriented images
   Flowers not facing front are removed, as it's not a easy task to identify flowers from the back, even to human.
3. Removed non-identifiable images
   Flower images such as landscapes, closeups, clusters are also removed. Flower species *astilbe* is removed entirely due to the nature that they all came in clusters, which is hard to be identified by the model.
4. Train test split
   Split the images into train, test, validation sets using python package *splitfolders*, with a ratio of 0.7, 0.15, 0.15 respectively.
5. Image Resizing and Random Transformation
   Images are loaded incrementally to our model using ImageDataGenerator, in which resizing and random transformation were applied to make our model more robust. Note that random transformation doesn't apply to Logistic Regression.

After the above cleaning process, we end up with 14070 images



MODELING AND EVALUATION

In total, 3 different deep learning models were established, including a baseline Logistic Regression Model, CNN model and CNN – VGG16 transfer learning model. The below table would summarize all of them in training and evaluation stages:

Model	**Logistic Regression**	**CNN**	**CNN -VGG16**
Hyperparameters	Saga, l1 penalty, C=5	30 epochs, ES on val_loss, Adam opt	60 epochs, Adam opt, lr 0.002
Total nodes	NA	2,115,375	14,714,688
Time to Train	1 h 35 min	57 min	4 h 7 min
Train Accuracy	84.31%	76.29%	77.26%
Val/Test Accuracy	38.61%	73.70%	79.91%
Evaluation	Not successful in examining the images and predict outcome pixels by pixels	Potentially higher accuracy with more layers and nodes	Potentially higher accuracy as train acc < val acc if given longer training time/faster learning rate

CONCLUSION

It appears that Logistic Regrsssion model struggles a lot with such a task. We are seeing serious overfitting to our data. A possible assumption was that our image data are too generalized and include a lot of irrelevant patterns or objects (such as a hand holding a flower or flowers on a cake). Such patterns are hard to be analyzed by Logistics Regression, as it's looking at the images pixels by pixels and assign weights to each one of them.

Another thing to point out is that it takes Logistic Regression around 50 min to finish the training. This is due to the fact that there are simply too many pixles, 64 * 64 = 4096 columns of data to be processed.

On the other hand, CNN and CNN – VGG 16 are more advanced in dealing with such a task. Both CNN models try to shrink our images and find the patterns (lines, curves, shapes, colors) regions by regions. They are more capable of identifying decisive elements in images. Of these two models, our CNN achieves 77.30% accuracy at the risk of overfitting, while CNN – VGG 16 is expected to be holding more potential as train_acc is still smaller than val_acc. We will use CNN – VGG 16 in our prediction

UI as this gives more accurate results. Also, it's also obvious to see that how a well-researched and developed model can outperform our model. There are more improvements to be done to our CNN model.

NEXT STEPS

We can use the CNN – VGG 16 model as the core of our prediction. However, 80% accuracy is not close to be released yet. Next steps would be further clean the dataset. Also, if needed, more images will be incorporated in our dataset to compensate for imbalanced categories. In the meantime, our prediction UI can be further improved to include more botanical information through web scraping and output similar images. Finally, we will use streamlit to establish a web app and put it into production.