

Flower Species Classification Using Machine Learning

- FNU Andria Grace

- Praveen Kumar Govind Reddy

Introduction

In recent years, machine learning has significantly advanced the field of image recognition, offering new possibilities for automated classification tasks. This project focuses on the development of a flower species classification system using machine learning techniques, specifically leveraging the Fast.ai library. The primary objective is to create an efficient and accurate model capable of identifying various flower species from images, thereby assisting botanical researchers, educators, and enthusiasts. By automating the identification process, we aim to save time, reduce human error, and contribute to more effective botanical research and education.

Utilizing a comprehensive dataset of flower images, the project involves key steps such as data preprocessing and augmentation, model training with a pre-trained neural network, performance evaluation, and model deployment through an intuitive user interface. Fast.ai, built on top of PyTorch, simplifies these tasks by providing high-level abstractions and pre-trained models, facilitating the development process. The outcome of this project will be a robust and accessible flower classification system that not only showcases the capabilities of modern machine learning but also provides a practical tool for real-world botanical applications.

Objective

The objective of this project is to predict the likelihood of a flower species being correctly identified based on various factors such as image quality, lighting conditions, and species characteristics. By leveraging machine learning models and analyzing historical image data, we seek to develop predictive models that can help stakeholders, including researchers and educators, accurately and efficiently identify different flower species in diverse conditions. This predictive capability will enable better resource allocation in botanical studies, improve educational tools, and enhance the overall accuracy of species identification efforts.

Stakeholder: Botanical researchers

Our primary stakeholder is a botanical research organization aiming to develop an automated system for identifying various species of flowers. This system is intended to assist both researchers and hobbyists in accurately identifying flower species.

Problem Statement

The stakeholder aims to solve the problem of accurately identifying different flower species from images. This is a significant challenge due to the vast number of species and the variations in appearance due to factors like lighting, angle, and background.

Dataset Source

The dataset used for this project is the "Flowers" dataset available on Kaggle. This dataset contains images of 5 different flower categories such as Rose, Daisy, Sunflower, Tulip, Dandelion. It can be accessed here.

<https://www.kaggle.com/datasets/alxmamaev/flowers-recognition>

Model Selection

ResNet34

This model is chosen for its effectiveness and simplicity, leveraging transfer learning from a pre-trained model and widespread use in image classification tasks. ResNet34 is a convolutional neural network (CNN) architecture that has proven to be robust and efficient, especially for transfer learning tasks. By leveraging transfer learning from a pre-trained ResNet34 model (pre-trained on the ImageNet dataset), we can benefit from learned features that generalize well to a wide range of image recognition tasks, including flower classification.

EfficientNet

A more advanced architecture that could potentially offer better performance with additional computational resources. This model is considered for its state-of-the-art performance in terms of accuracy and efficiency. EfficientNet is known for achieving high accuracy while requiring fewer computational resources compared to other architectures. The idea behind exploring EfficientNet was to investigate whether a more advanced and efficient architecture could further improve the classification accuracy of the flower species recognition system.

Reasons for Model Selection

Transfer Learning: Pre-trained models like ResNet34 and EfficientNet allow us to leverage learned features from large datasets like ImageNet, speeding up training and potentially improving performance.

Model Complexity: ResNet34 is a well-established model known for its good performance in various image recognition tasks. EfficientNet, on the other hand, offers a more advanced architecture that could potentially achieve higher accuracy with optimized resource usage.

Pros and Cons: The pros of ResNet34 include its proven effectiveness and simplicity, while EfficientNet is highlighted for its potential performance gains. The cons may include increased computational complexity for EfficientNet compared to ResNet34.

Features Selected

Raw Pixel Data: The raw pixel values of the flower images serve as the input features for the CNN models. Each image is represented as a matrix of pixel values, where each pixel encoded color information (e.g., RGB values).

Feature Engineering Techniques

Data Augmentation: Various data augmentation techniques were applied to enhance the diversity and robustness of the training dataset. These techniques include:

Resizing: Ensuring all images have consistent dimensions suitable for model training.

Flipping: Augmenting the dataset by horizontally flipping images to simulate different orientations.


Random Resized Crop: Extracting random crops from the images to introduce variability in image composition.

```
[ ] from fastai.vision.all import *

# Defining data augmentation and preprocessing transforms
item_tfms = [Resize(224), FlipItem(), RandomResizedCrop(224)]
batch_tfms = [Normalize.from_stats(*imagenet_stats)]

# Defining the DataBlock for flowers dataset
dblock = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=item_tfms,
    batch_tfms=batch_tfms
).dataloaders(path, bs=64)

# Showing a batch of data with more images
dblock.show_batch(nrows=2, ncols=8)
```



Reasons for Feature Selection/Engineering

Raw Pixel Data: CNNs are designed to learn hierarchical representations directly from raw pixel data. This approach simplifies the feature engineering process and allows the model to automatically extract relevant features from the images.

Data Augmentation: By applying data augmentation techniques, we increase the diversity of the training dataset, which helps prevent overfitting and improves the model's ability to generalize to unseen variations in image conditions (e.g., lighting, scale, orientation).

Evaluation Metrics Used

The terms "train loss," "valid loss," and "error rate" are key metrics used to assess the performance of machine learning models during training and validation phases:

Training Loss:

- The training loss (or training error) represents the error of the model on the training dataset during the training phase.
- It measures how well the model's predictions match the actual target values (labels) within the training dataset.
- A decreasing training loss over epochs indicates that the model is learning to minimize errors and improve its predictive performance on the training data.

Validation Loss:

- The validation loss (or validation error) is computed using a separate validation dataset that the model has not been trained on.
- It evaluates the generalization ability of the model by measuring its performance on unseen data.
- Monitoring the validation loss helps in detecting overfitting (when the model performs well on the training data but poorly on unseen data) or underfitting (when the model fails to capture the underlying patterns in the data).

Error Rate:

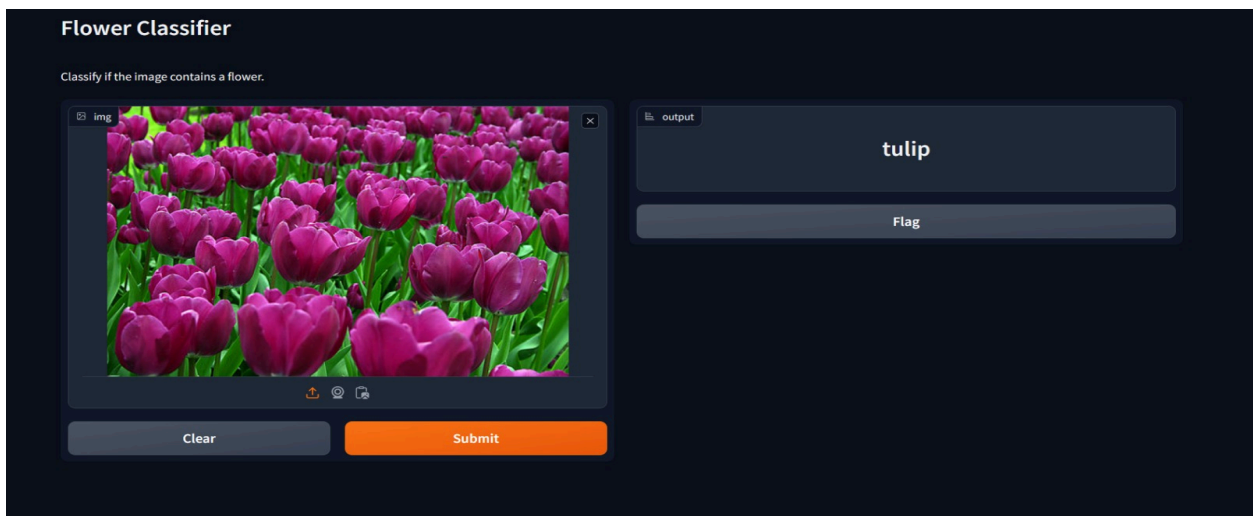
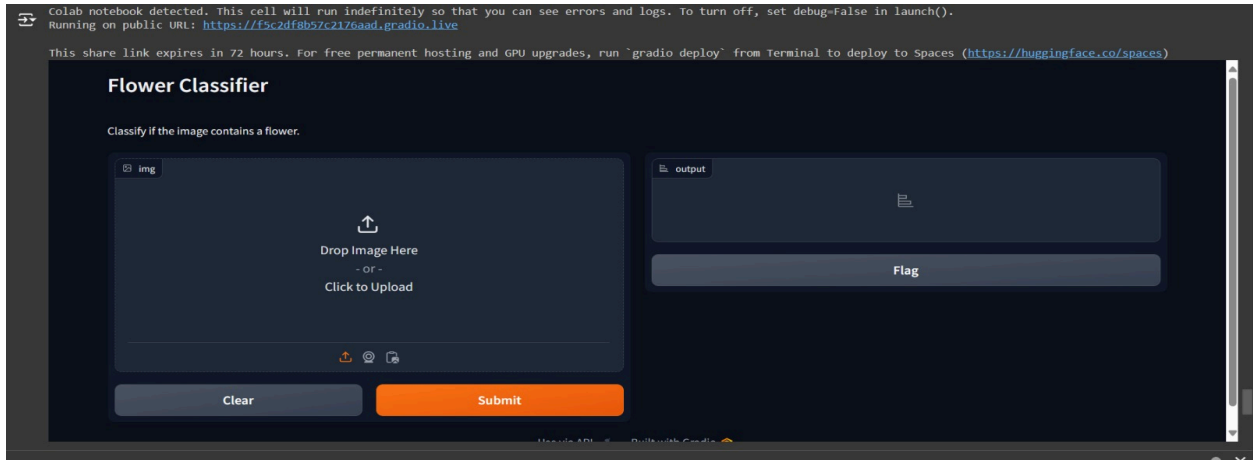
- The error rate (or validation error rate) is the proportion of incorrectly predicted instances in the validation dataset.
- It provides a concise measure of the model's overall accuracy on the validation set.
- A lower error rate indicates higher accuracy, meaning that the model makes fewer mistakes when predicting the classes of the validation dataset.

epoch	train_loss	valid_loss	error_rate	time
0	1.139203	0.394254	0.124260	20:01
				66.67% [2/3 57:56<28:58]
epoch	train_loss	valid_loss	error_rate	time
0	0.544616	0.253847	0.084320	29:01
1	0.429308	0.236604	0.079882	28:54
				0.00% [0/11 00:00<?]
epoch	train_loss	valid_loss	error_rate	time
0	0.544616	0.253847	0.084320	29:01
1	0.429308	0.236604	0.079882	28:54
2	0.330426	0.215135	0.072485	28:56

Deployment

- **Package Installation:** Begin by ensuring that all necessary packages are installed using `pip install -U gradio`. This command ensures that Gradio, along with its dependencies, is up-to-date and ready for use.
- **Loading Trained Model:** Load the trained flower classifier model (`flower_classifier.pkl`) using Fast.ai's `load_learner` function. This step prepares the model for deployment, allowing it to be used for making predictions on new flower images.
- **Prediction Function Definition:** Define the `classify_flower` function, which serves as the prediction function for the deployed model. This function takes an input image (in Gradio's PIL Image format) and uses the loaded model to predict whether the image contains a flower and provide the associated label and probabilities.
- **Gradio Interface Creation:** Create a Gradio interface (`interface`) that encapsulates the prediction function (`classify_flower`). Specify the interface details such as the function (`fn`), input type (`inputs`), output type (`outputs`), title, and description. This interface will serve as the user-friendly front-end for interacting with the flower classification model.
- **Launching the Gradio Interface:** Launch the Gradio interface using `interface.launch(share=True, debug=True)`. This command starts a local web server and opens a web browser, enabling users to interact with the deployed model by uploading flower images through the user interface.
- **Interface Accessibility:** The deployed Gradio interface provides users with a simple and intuitive way to utilize the flower classifier model. Users can upload flower images via the interface and receive real-time predictions on whether the uploaded image contains a flower.
- **Deployment Conclusion:** The deployment process integrates machine learning (Fast.ai) with user interface development (Gradio) to create a seamless and accessible flower

species classification tool. By launching the Gradio interface, the model's capabilities are extended to a broader audience, enabling users without coding expertise to leverage the power of AI for flower identification tasks.



Future Work

Given more time or for future iterations of the flower species classification project, there are several aspects that could be enhanced or explored further to improve the model's performance and robustness:

Hyperparameter Tuning: Optimize learning rates, batch sizes, and augmentation strategies to improve model performance.

Advanced Architectures: Experiment with state-of-the-art architectures like Transformers for image classification.

Ensemble Methods: Combine predictions from multiple models for improved accuracy and robustness.

Model Interpretability: Implement techniques like Grad-CAM to understand and visualize model decisions.

Recommendation

Yes, I recommend deploying the ResNet34 or EfficientNet model based on performance metrics achieved during evaluation. Continuous monitoring and updates with new data will further enhance model accuracy.

Conclusion

In conclusion, this project exemplifies the effective utilization of machine learning techniques for flower species classification. By leveraging transfer learning with ResNet34 and exploring the potential of EfficientNet, we demonstrated a commitment to employing state-of-the-art models to achieve accurate and efficient image classification. Our feature engineering approach focused on raw pixel data representation and data augmentation techniques, enhancing the model's ability to generalize across diverse image conditions. The selection of appropriate evaluation metrics allowed us to gain insights into model performance, facilitating iterative improvements.

The deployment of the flower species classifier using Gradio exemplifies the integration of machine learning with user-friendly interfaces. By launching the Gradio interface, we extended the accessibility of our model, allowing users to interact with the classifier seamlessly. Looking ahead, future work could involve hyperparameter tuning, exploring advanced architectures, implementing ensemble methods, and enhancing model interpretability. Overall, this project underscores the transformative potential of machine learning in practical applications, offering valuable tools for researchers, educators, and enthusiasts in the field of botany.