Flower Identification

1. Data:

There is an abundance of data on flowers on the internet. Nonetheless, I will be using a Kaggle dataset with over 7000 unlabeled images and 4 folders with images of different sizes each. If the data is not enough, Kaggle also provides other flower datasets which can be added on top to this dataset. To further obtain more data and produce a more accurate model the dataset can be augmented.

Kaggle Dataset: https://www.kaggle.com/competitions/tpu-getting-started

Kaggle reference notebook: https://www.kaggle.com/code/georgezoto/computer-vision-petals-to-the-metal#Step-1:-Imports

2.Meathod

I will be using a deep learning to create the classification model. For all my models I will be using a sequential model as it is the most common deep learning model used. Furthermore, I will processes the data using TensorFlow and Keras to use the available TPUs on Kaggle to speed up the process. Moreover, I will be applying transfer learning techniques and using common Convolutional neural networks that have been known to produce highly accurate models.

Model 1: Used a sequential deep learning model using MobileNetV2. While referencing the Kaggle notebook Vgg16 and MobileNetV2 seemed to produce the best results for this data set; however, in my first run I did not get any good results while using Vgg16, thus I tried MobileNetV2 with drop-out. MobileNetV2 with drop-out gave me decent results with a f1 score of 0.40, and a precision score of 63%.

Model2: For model 2 I tried using a bit more complex model known as inceptionresnetV2. This model is highly accurate; however, because it is more complex it tends to overfit data. Nonetheless, for this dataset I obtained optimal results using inceptionresnetV2, with an f1 score of 0.90 and a precision score of 90%.

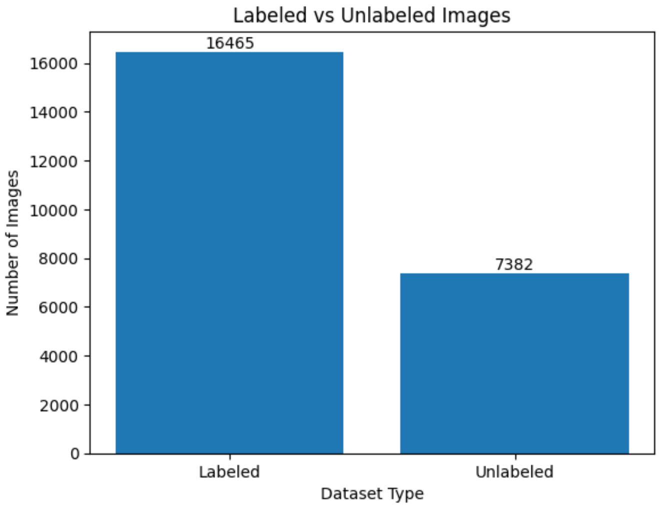
3.Data Cleaning

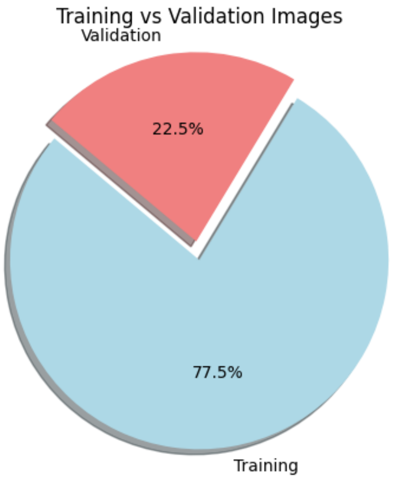
The data cleaning process for this project was fairly simple since there were no discrepancies in the data. Usually, the data cleaning process consists of resizing data, squishing data, and data augmentation. The data that I used was divided into 4 folders that contained flower images in different sizes: 192x192, 224x224, 331x331, 512x512. For this project I used the images in the 512x512 folder.

The data for this project was also neatly divided into training data and testing data, labeled data and unlabeled data respectively.

4.EDA

There was a much higher amount of labeled data than unlabeled data. Usually, it is much more difficult to find labeled data. Yet, I used data augmentation techniques to increase the amount of image data available and add further variation to my data. I randomly flipped my data, and I added a random contrast so the algorithm could get used to seeing variances in the data.





5. Algorithm and Deep Learning

I chose to work with the TensorFlow Keras library for training my image classification model. I tried different parameters including adding and removing drop out. I tested two algorithms on the 512x512 image data. I used the F1 score as an accuracy metric.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

$$precision = \frac{TP}{TP + FP}$$

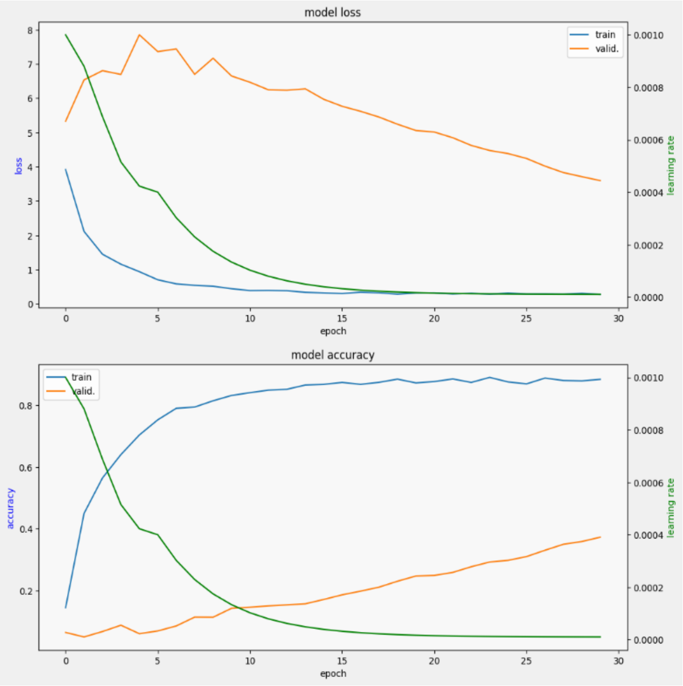
$$recall = \frac{TP}{TP + FN}$$

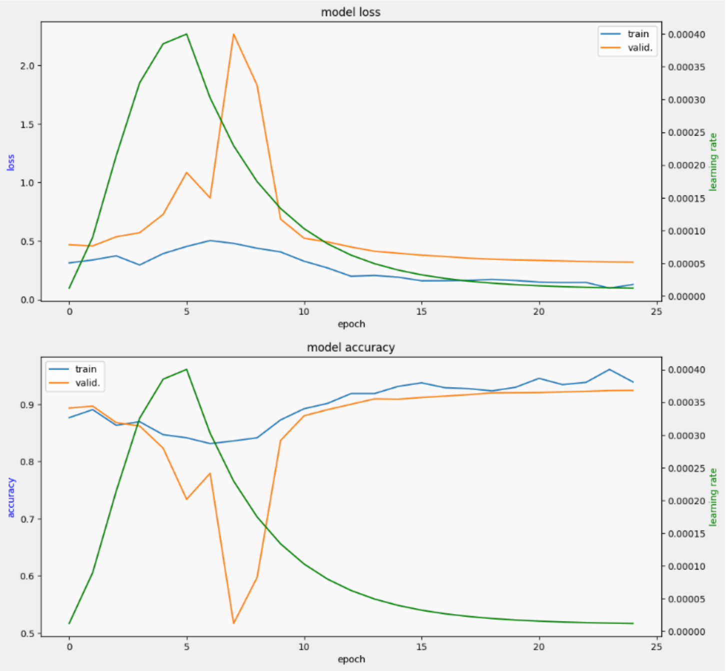
6. Results and Choosing the Best Model.

The best model was model 2 with an F1 score of 0.90 and precision of 91%. At first there was high model loss. Right after, accuracy dipped, and then slowly increases until it became steady. Model loss was inversely proportional to accuracy. The best model had the least model loss and the highest accuracy percentage.

- **Model Loss:** Compares target and predicted output values.

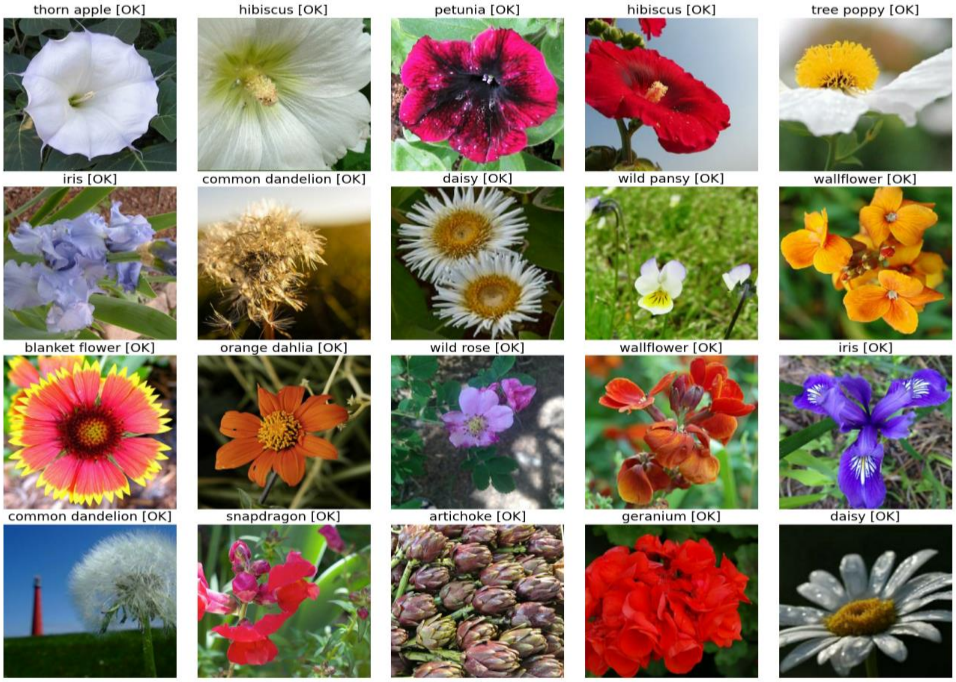
$$J(w^T, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

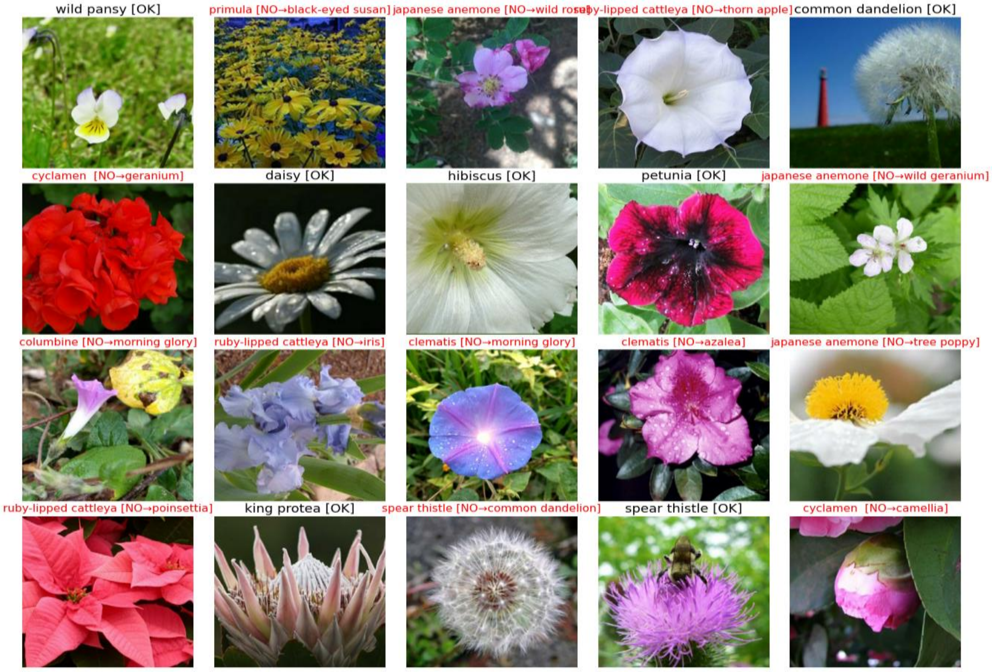




7.Predictions

In the final Part of this project, I called the validation set to test my models accuracy. I used a function I defined to print the images with their predicted labels. The difference in model accuracy is highly visible, as model 1 had mislabeled images with red labels, and model 2 had no images that were mislabeled.




wild pansy [OK]
primula [NO→black-eyed susan]
japanese anemone [NO→wild rose]
ruby-lipped cattleya [NO→thorn apple]
common dandelion [OK]
cyclamen [NO→geranium]
daisy [OK]
hibiscus [OK]
petunia [OK]
japanese anemone [NO→wild geranium]
columbine [NO→morning glory]
ruby-lipped cattleya [NO→iris]
clematis [NO→morning glory]
clematis [NO→azalea]
japanese anemone [NO→tree poppy]
ruby-lipped cattleya [NO→poinsettia]
king protea [OK]
spear thistle [NO→common dandelion]
spear thistle [OK]
cyclamen [NO→camellia]

8. Future Recommendations

- In the future I would like to expand this recognition system to identify seeds and match them to their flower.

- To further improve the best model, I would recommend experimenting on the parameters, especially the pooling parameter. Changing the average pooling parameter to max pooling could increase the models' F1 and precision score.
- Finally, this model can be used in the field of ecology to help identify unidentified flower species. This is especially helpful when trying to prevent or control invasive species that may be accidentally introduced.

9. Credits

Thanks to George Zoto for his in depth Kaggle notebook explaining the many ways the model can be made and optimized.