PlantDex

Matthew Belcher, Kristal Concepcion, Jasmine Leal, Chloe Valverde

SYNOPSIS

We created an image recognition program that will be able to predict whether plants are poisonous or safe segments, it will run the uploaded picture through the database and with confident probability pair it with the correct image and name of the plant and then predict whether that plant is toxic or non toxic for human contact.

Our target audience are botanists and horticulturists so they can easily identify the plants they want to research on. In addition to this, some side target audiences are florists, home gardeners, hikers, and pet owners.

RESEARCH QUESTIONS

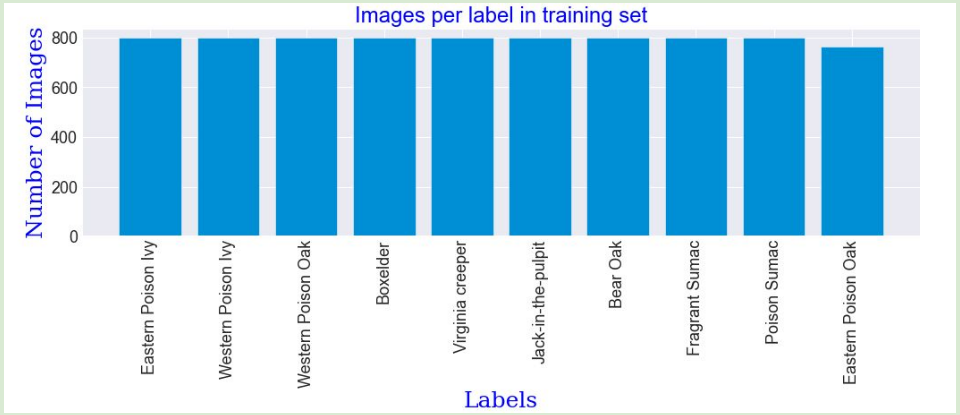
- How accurately can we predict the plant type?
- What kind of model is best used/how to improve our model?
- Can we determine if they are poisonous?

DATA DESCRIPTION

The dataset that was used is of 10,000 images of 8 different plant types.

We also used a separate dataset that provided the scientific name of the plant, toxicity levels, herbarium category, species level, and slang name.

From this data set we were able to clean up the CSV's and narrow down to Slang name, toxicity level, and path. This helped with improving the time needed to run the prediction.



PROCESS AND METHODS

Data Process

Data acquisition - finding a diverse dataset with necessary requirements (picture, name, toxicity)

Data cleaning - cleaning the csv of unnecessary information (scientific name, species label)

Data manipulation - separating the data into the X_train, y_train, X_test, and y_test. These data frames consisted of both the "common name", the path to specific photo, toxicity level

Once the data was organize and reading for modeling we created a few different methods.

Code Process

makeGen method - this method generates the pictures using tensorflow keras ImageDataGenerator

showImageSample - display a 25x25 grid of random images pulled from the X_train generator

makeModel - creates and trains the models using tensorflow keras

LRask - allows the user to input the number of times they want the code ran, since we used the transfer learning method the more times the code iterate the higher the f1score

trPlot - plots out the results (loss, accuracy, and f1score)

Transfer Learning

What is Transfer Learning?

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It focuses on storing knowledge gained while solving one problem and applying it to the next problem

All you need to do is create a class that iterates through the training data whenever you call it, however many times you call it. And it will use the knowledge gained from the previous results and apply it to the next iteration.

Why did we choose this?

For CNN you need to more preprocessing of the images, but with transfer learning you only need to little processing of dataset.

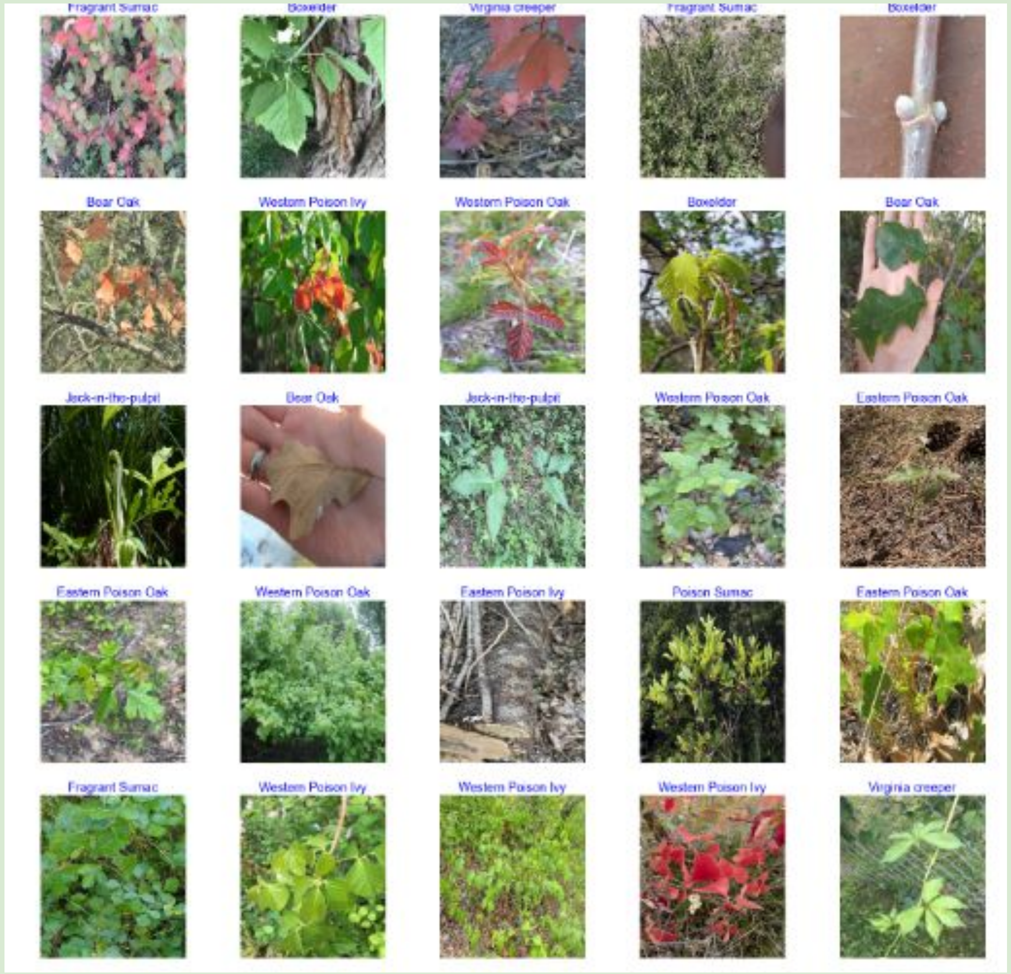
Training a cnn takes a long time, especially with huge datasets, this was a problem we encountered since our data set contained 10,000 images

Conventional machine learning has been traditionally designed to work in isolation. The algorithms are trained to solve specific tasks. While transfer learning is the idea of overcoming the isolated learning problem

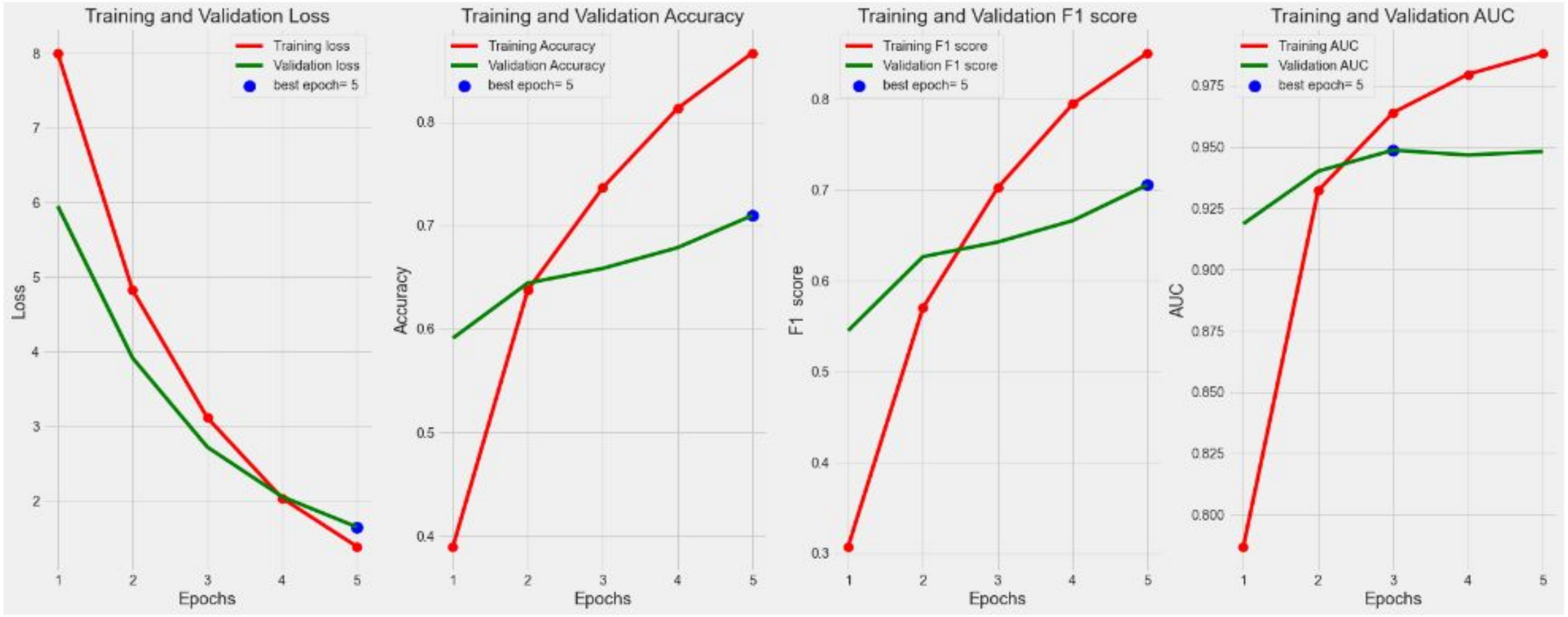
SUMMARY REPORT

GRID GRAPH

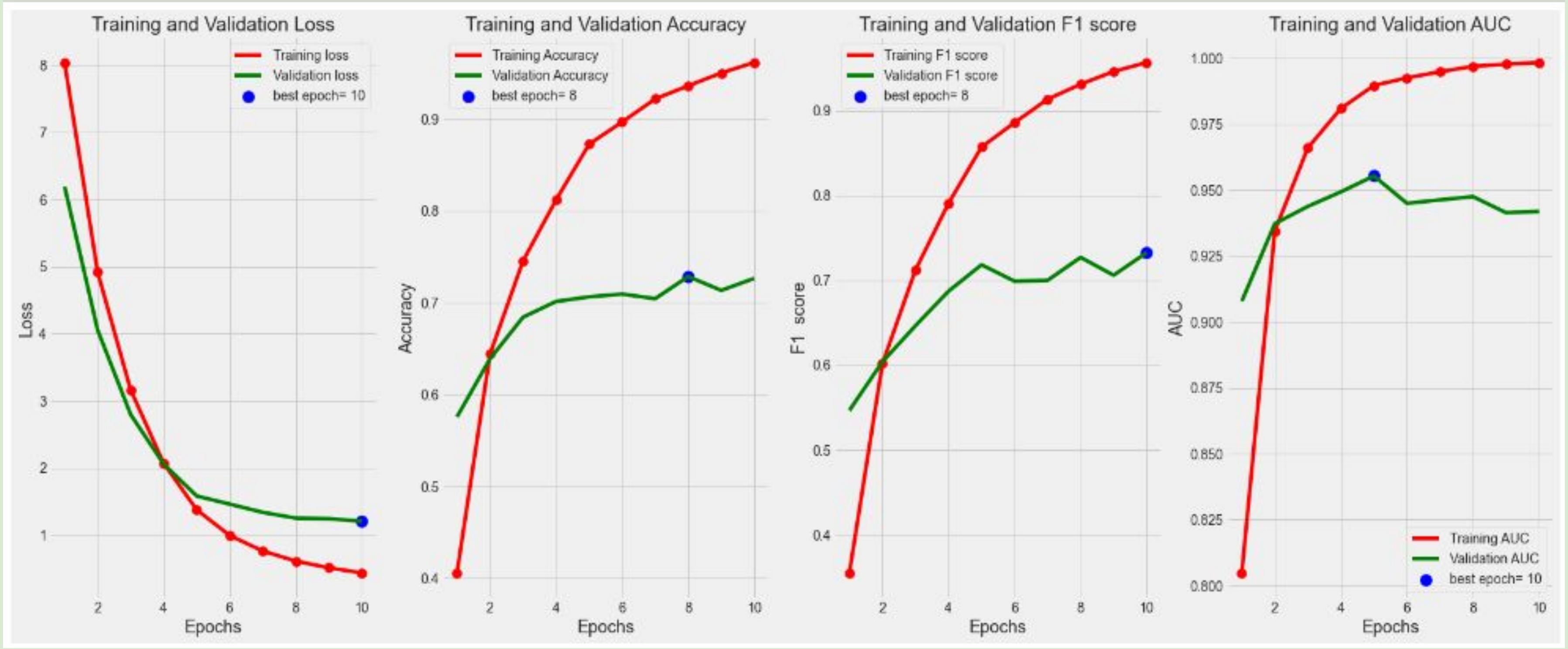
25 sample pictures taken from X-train and X-test code, called from excel herbarium with over 10,000 files



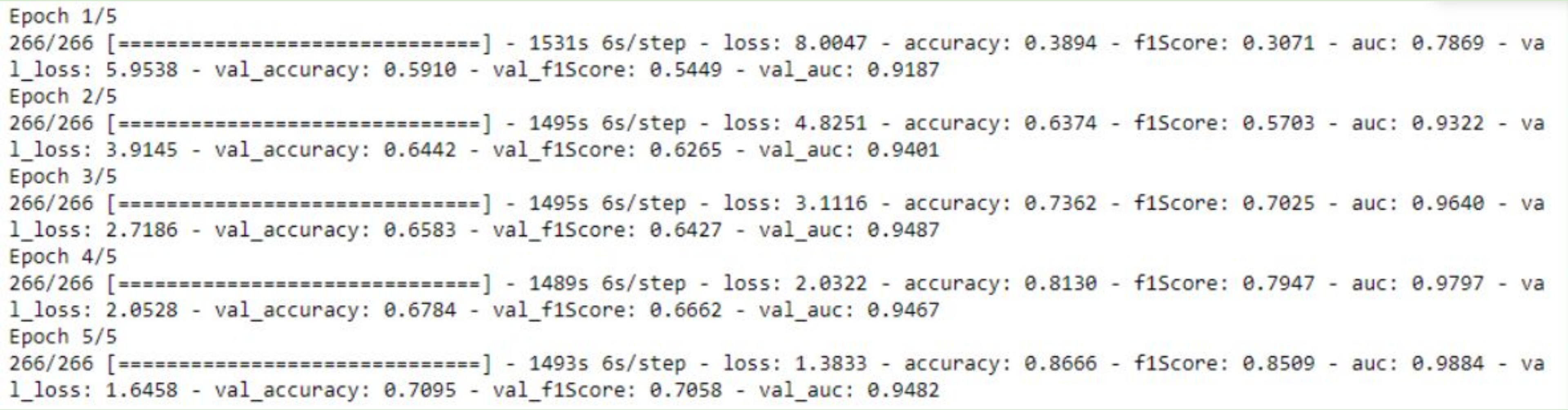
Training and Validation Loss (5 Epochs)



Training and Validation Loss (10 Epochs)



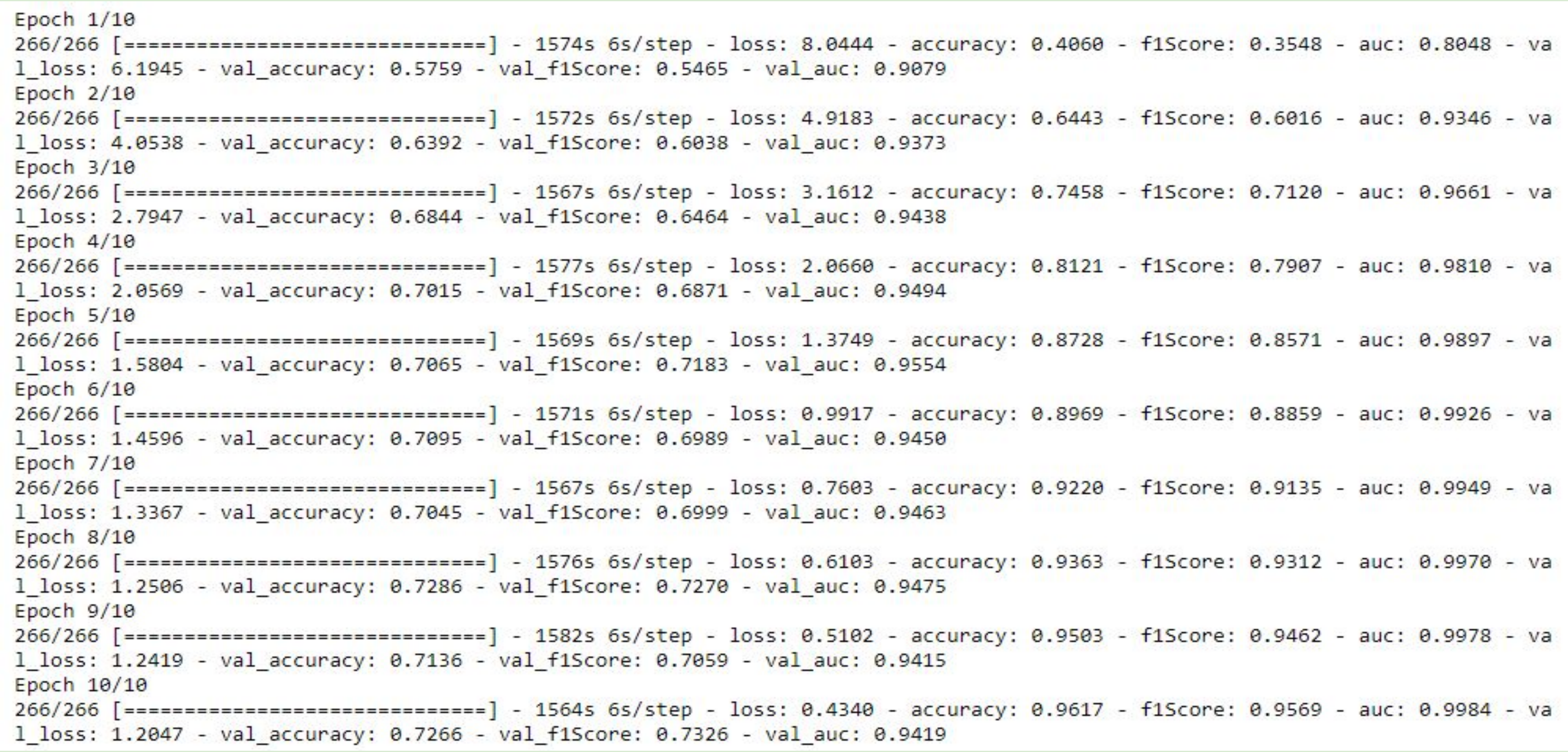
FIRST 5 Epochs



```
Epoch 1/5
266/266 [==============================] - 1531s 6s/step - loss: 8.0047 - accuracy: 0.3894 - f1Score: 0.3071 - auc: 0.7869 - va
l_loss: 5.9538 - val_accuracy: 0.5910 - val_f1Score: 0.5449 - val_auc: 0.9187
Epoch 2/5
266/266 [==============================] - 1495s 6s/step - loss: 4.8251 - accuracy: 0.6374 - f1Score: 0.5703 - auc: 0.9322 - va
l_loss: 3.9145 - val_accuracy: 0.6442 - val_f1Score: 0.6265 - val_auc: 0.9401
Epoch 3/5
266/266 [==============================] - 1495s 6s/step - loss: 3.1116 - accuracy: 0.7362 - f1Score: 0.7025 - auc: 0.9640 - va
l_loss: 2.7186 - val_accuracy: 0.6583 - val_f1Score: 0.6427 - val_auc: 0.9487
Epoch 4/5
266/266 [==============================] - 1489s 6s/step - loss: 2.0322 - accuracy: 0.8130 - f1Score: 0.7947 - auc: 0.9797 - va
l_loss: 2.0528 - val_accuracy: 0.6784 - val_f1Score: 0.6662 - val_auc: 0.9467
Epoch 5/5
266/266 [==============================] - 1493s 6s/step - loss: 1.3833 - accuracy: 0.8666 - f1Score: 0.8509 - auc: 0.9884 - va
l_loss: 1.6458 - val_accuracy: 0.7095 - val_f1Score: 0.7058 - val_auc: 0.9482
```

When we ran the the first five epochs, we note that the initial accuracy of our first epoch is about 30%. This is based on our f1Score which determines how accurate the program is functioning in terms of percentage. We note that as we run each epoch, we end up increasing the accuracy of program. We also note that it isn’t until the fifth epoch when we have a program accuracy of about 85%.

NEW TEST WITH 10 EPOCHS



```
Epoch 1/10
266/266 [==============================] - 1574s 6s/step - loss: 8.0444 - accuracy: 0.4060 - f1Score: 0.3548 - auc: 0.8048 - va
l_loss: 6.1945 - val_accuracy: 0.5759 - val_f1Score: 0.5465 - val_auc: 0.9079
Epoch 2/10
266/266 [==============================] - 1572s 6s/step - loss: 4.9183 - accuracy: 0.6443 - f1Score: 0.6016 - auc: 0.9346 - va
l_loss: 4.0538 - val_accuracy: 0.6392 - val_f1Score: 0.6038 - val_auc: 0.9373
Epoch 3/10
266/266 [==============================] - 1567s 6s/step - loss: 3.1612 - accuracy: 0.7458 - f1Score: 0.7120 - auc: 0.9661 - va
l_loss: 2.7947 - val_accuracy: 0.6844 - val_f1Score: 0.6464 - val_auc: 0.9438
Epoch 4/10
266/266 [==============================] - 1577s 6s/step - loss: 2.0660 - accuracy: 0.8121 - f1Score: 0.7907 - auc: 0.9810 - va
l_loss: 2.0569 - val_accuracy: 0.7015 - val_f1Score: 0.6871 - val_auc: 0.9494
Epoch 5/10
266/266 [==============================] - 1569s 6s/step - loss: 1.3749 - accuracy: 0.8728 - f1Score: 0.8571 - auc: 0.9897 - va
l_loss: 1.5804 - val_accuracy: 0.7065 - val_f1Score: 0.7183 - val_auc: 0.9554
Epoch 6/10
266/266 [==============================] - 1571s 6s/step - loss: 0.9917 - accuracy: 0.8969 - f1Score: 0.8859 - auc: 0.9926 - va
l_loss: 1.4596 - val_accuracy: 0.7095 - val_f1Score: 0.6989 - val_auc: 0.9450
Epoch 7/10
266/266 [==============================] - 1567s 6s/step - loss: 0.7603 - accuracy: 0.9220 - f1Score: 0.9135 - auc: 0.9949 - va
l_loss: 1.3367 - val_accuracy: 0.7045 - val_f1Score: 0.6999 - val_auc: 0.9463
Epoch 8/10
266/266 [==============================] - 1576s 6s/step - loss: 0.6103 - accuracy: 0.9363 - f1Score: 0.9312 - auc: 0.9970 - va
l_loss: 1.2506 - val_accuracy: 0.7286 - val_f1Score: 0.7270 - val_auc: 0.9475
Epoch 9/10
266/266 [==============================] - 1582s 6s/step - loss: 0.5102 - accuracy: 0.9503 - f1Score: 0.9462 - auc: 0.9978 - va
l_loss: 1.2419 - val_accuracy: 0.7136 - val_f1Score: 0.7059 - val_auc: 0.9415
Epoch 10/10
266/266 [==============================] - 1564s 6s/step - loss: 0.4340 - accuracy: 0.9617 - f1Score: 0.9569 - auc: 0.9984 - va
l_loss: 1.2047 - val_accuracy: 0.7266 - val_f1Score: 0.7326 - val_auc: 0.9419
```

BRINGING IT AROUND TOWN, AROUND TOWN

The plants are green, edible, and if not edible, then edible only once.

In conclusion the overall prediction accuracy is up to 96.17 % with 95.69 F1, improving vastly each time the model is run due to the transfer method. Some key takeaways were the trial and errors of different methods and the importance of run times. We hope to in the future to scale this by expanding the dataset and enhancing accuracy and efficiency.

Resources/Citations

- https://www.kaggle.com/code/aderezapahlevi/starter-flowers-recognition-7123e64e-8/data
- https://www.kaggle.com/code/tarunpaparaju/plant-pathology-2020-eda-models/notebook
- https://www.kaggle.com/code/rajmehra03/flower-recognition-cnn-keras/notebook
- https://www.kaggle.com/datasets/hanselliott/toxic-plant-classification/code
- https://machinelearningmastery.com/train-to-the-test-set-in-machine-learning/
- https://plants.usda.gov/home
- https://plantdatabase.uconn.edu/

Github repo
https://github.com/kconcepcion/flower-identifier-and-almanac