Identifying Poisonous Plants Using Deep Learning

Qurat Ul Ain Syeda
MSIS 2626: Deep Learning
Fall 2021

Agenda

01 Defining the Problem

The motivation behind choosing this topic

02 Data Collection

Using Plant Clef Dataset & Google Images

03 Architectures

Simple CNN, Mini-Xception, Transfer Learning & HyperParameter-tuning with KerasTuner

04 Conclusion

Comparison of results & final model selection

Motivation

Background

- ❖ Poisonous plants can cause an allergic reaction in 80-90% of adults
- ❖ Poisonous plants (eg. poison ivy, poison oak, poison sumac) release an oil called urushiol which can cause:
  - ➢ Rashes, blisters, bumps are non-threatening
  - ➢ Oil can cause lung irritation if inhaled
  - ➢ Advisable to see a doctor if too close to eyes or widespread

Use Cases

- ❖ Integrate into nature-related mobile application
- ❖ Eg. hiking trails app, bird watching app, flower/plant identifier app
- ❖ Point, click, and identify functionality

Data Collection

Dataset divided into poisonous and non-poisonous plants

- ❖ Poisonous plants: Downloaded approx. 100 images for each type of plant. Total: 270 images
- ❖ Used PlantCLEF 2015 Annotated Dataset: Consists of over 1000 images submitted by the users of the mobile application Pl@ntNet. Total: 270 images

MODEL ARCHITECTURES

Simple CNN

Using multiple convolutional layers

Mini-Xception

Truncated version of Xception

Transfer Learning

Using Xception & ImageNet

Simple CNN - Workflow

01 Augment Images w/ ImageDataGenerator()

Use .flow_from_dirtectory() method to load images & apply image augmentation during training

02 Build, compile & train the model

Used multiple convolutional layers & RMSProp as an optimization algorithm.

03 Evaluate model performance & use to predict on test set

Goal is to maximize validation accuracy

Simple CNN - Model

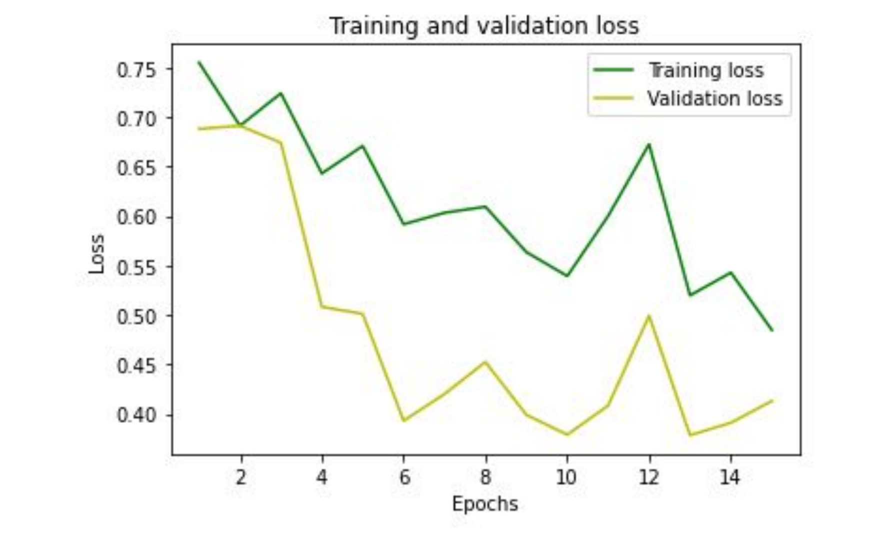
Model: "sequential_1"

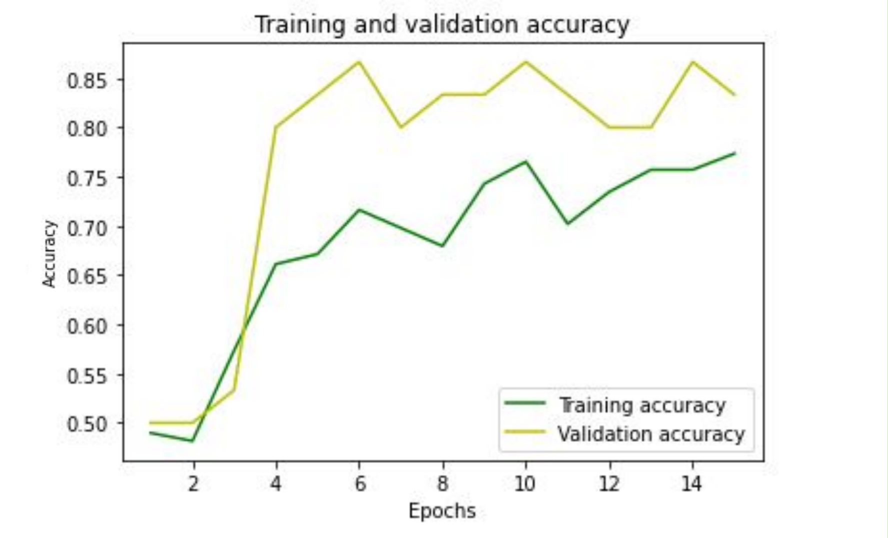
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 16)	448
max_pooling2d (MaxPooling2D)	(None, 99, 99, 16)	0
conv2d_1 (Conv2D)	(None, 97, 97, 32)	4640
max_pooling2d_1 (MaxPooling2	(None, 48, 48, 32)	0
conv2d_2 (Conv2D)	(None, 46, 46, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 23, 23, 64)	0
conv2d_3 (Conv2D)	(None, 21, 21, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 10, 10, 64)	0
conv2d_4 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_4 (MaxPooling2	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 512)	524800
dense_3 (Dense)	(None, 1)	513

Total params: 622,753
Trainable params: 622,753
Non-trainable params: 0

Simple CNN - Training Results

Validation accuracy greater than training due to size & possible difference in noise/variance in validation dataset



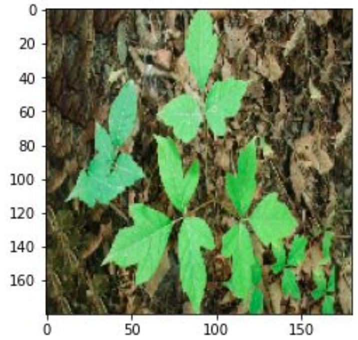


Training Loss = 0.485
Validation Loss = 0.413

Training Accuracy = 77.35%
Validation Accuracy = 83.33%

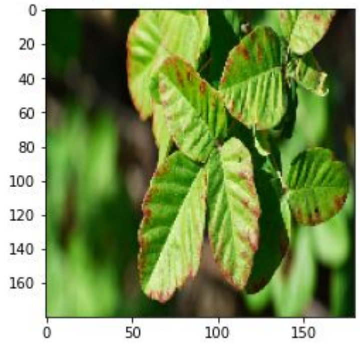
Simple CNN - Predicting on Test Set

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



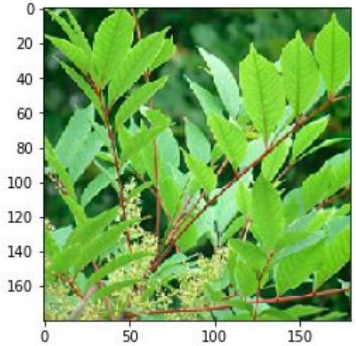
Poison Ivy

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



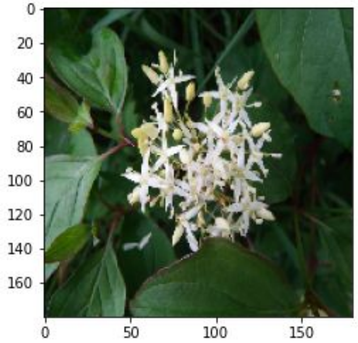
Poison Oak

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



Poison Sumac

This image is 100.00 percent non-poisonous and 0.00 percent poisonous.



Non-Poisonous Plant

Simple CNN - Hyperparameter Tuning

- KerasTuner is an easy-to-use, scalable hyperparameter optimization framework that solves the pain points of hyperparameter search
- We used the **RandomSearch()** algorithm to find the best hyperparameter values for our model with the goal of maximizing "validation accuracy"
- The hyperparameters we tuned:
  - # of neurons in the last hidden layer
  - Learning rate supplied to our RMSProp optimizer

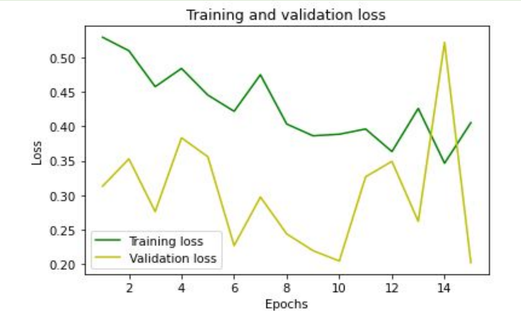
```
tuner.search_space_summary()
```

```
Search space summary
Default search space size: 2
units (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step': 128, 'sampling': None}
learning_rate (Choice)
{'default': 0.01, 'conditions': [], 'values': [0.01, 0.001, 0.0001], 'ordered': True}
```

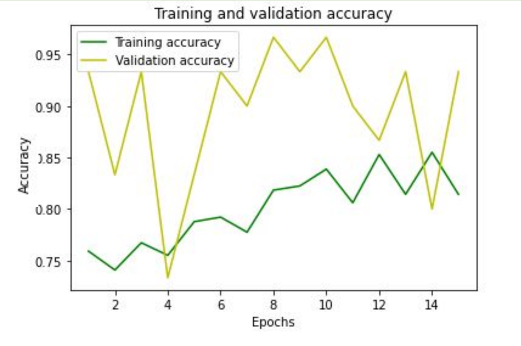
Hyperparameter Tuning Results

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 16)      448

 max_pooling2d (MaxPooling2D  (None, 99, 99, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 97, 97, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 48, 48, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 21, 21, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 8, 8, 64)          36928

 max_pooling2d_4 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 160)               164000

 dense_1 (Dense)             (None, 1)                 161

=================================================================
Total params: 261,601
Trainable params: 261,601
Non-trainable params: 0
```

```
Trial summary
Hyperparameters:
units: 160
learning_rate: 0.001
Score: 0.9111110965410868
```



Training Loss = 0.405
Validation Loss = 0.202



Training Accuracy = 81.43%
Validation Accuracy = 93.33%

Mini-Xception Model- Workflow

01 **Load & split dataset using image_dataset_from_directory() & augment images using Keras' data augmentation layers**

Performed an 80/20 split on dataset & used RandomFlip & RandomRotation layers

02 **Build, compile & train the model**

Uses pointwise convolution followed by a depthwise convolution

03 **Evaluate model performance & use to predict on test set**

Goal is to maximize validation accuracy

Xception - Image Augmentation

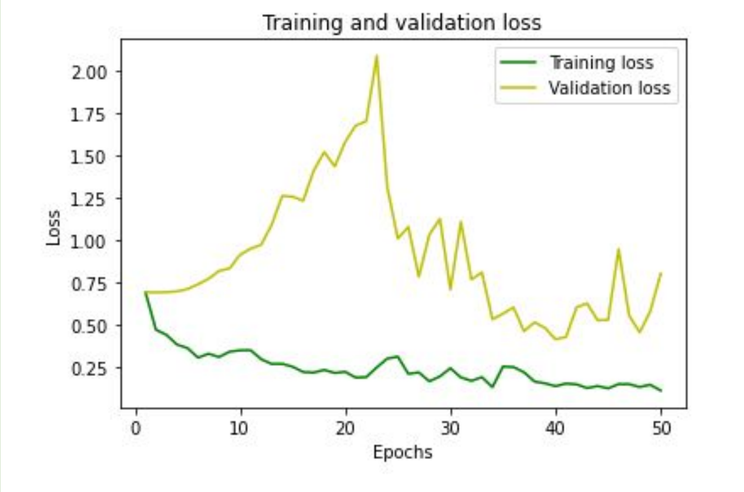
- Since we have a limited dataset, we used data augmentation layers.
- These layers apply random augmentation transforms to a batch of images.
- They are only active during training.
- We applied the following transformations:
  - *tf. keras.layers.RandomFlip(""horizontal_and_vertical")*
  - *tf.keras.layers.RandomRotation(0.2),*

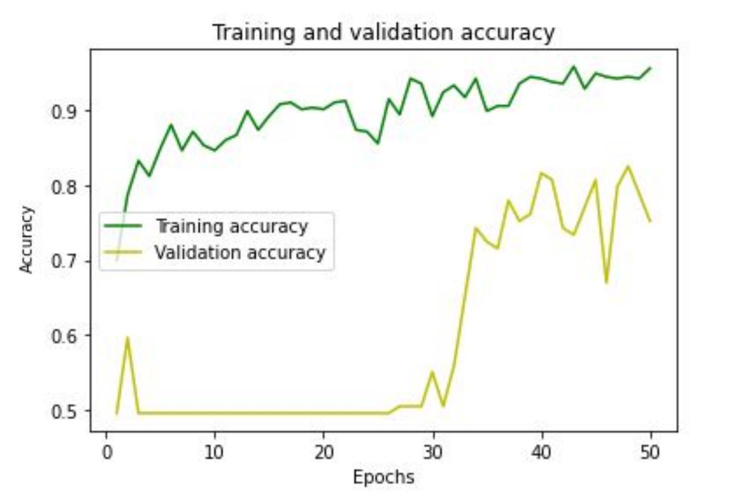
Xception - Model Summary (Partial)

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 180, 180, 3  0           []
                                )]

 sequential (Sequential)        (None, 180, 180, 3)  0           ['input_1[0][0]']

 rescaling (Rescaling)          (None, 180, 180, 3)  0           ['sequential[0][0]']

 conv2d (Conv2D)                (None, 90, 90, 32)   896         ['rescaling[0][0]']

 batch_normalization (BatchNorm  (None, 90, 90, 32)  128         ['conv2d[0][0]']
 alization)

 activation (Activation)        (None, 90, 90, 32)   0           ['batch_normalization[0][0]']

 conv2d_1 (Conv2D)              (None, 90, 90, 64)   18496       ['activation[0][0]']
```

```
 batch_normalization_10 (BatchN  (None, 6, 6, 1024)  4096        ['separable_conv2d_8[0][0]']
 ormalization)

 activation_10 (Activation)     (None, 6, 6, 1024)   0           ['batch_normalization_10[0][0]']

 global_average_pooling2d (Glob  (None, 1024)        0           ['activation_10[0][0]']
 alAveragePooling2D)

 dropout (Dropout)              (None, 1024)         0           ['global_average_pooling2d[0][0]'
                                                                 ]

 dense (Dense)                  (None, 1)            1025        ['dropout[0][0]']

==================================================================================================
Total params: 2,782,649
Trainable params: 2,773,913
Non-trainable params: 8,736
```

Xception Model - Training Results





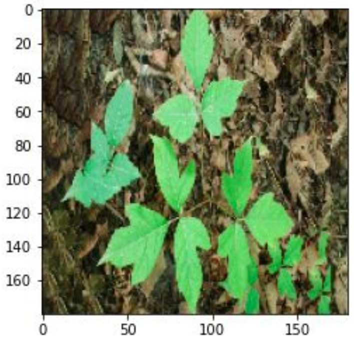
Training Loss = 0.114
Validation Loss = 0.803

Training Accuracy = 95.65%
Validation Accuracy = 75.23%

Validation accuracy is significantly lower than training due to increased model complexity & limited dataset

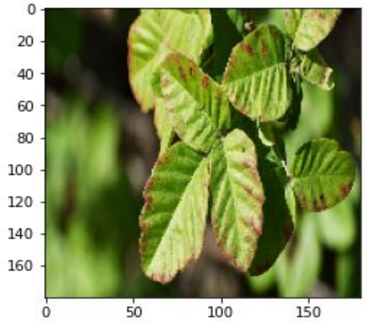
Mini-Xception - Predicting on Test Set

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



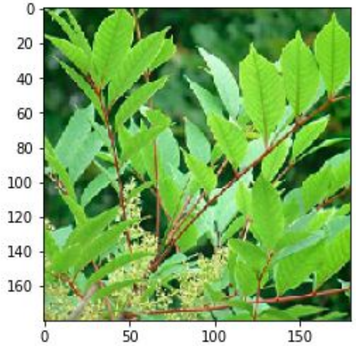
Poison Ivy

This image is 0.71 percent non-poisonous and 99.29 percent poisonous.



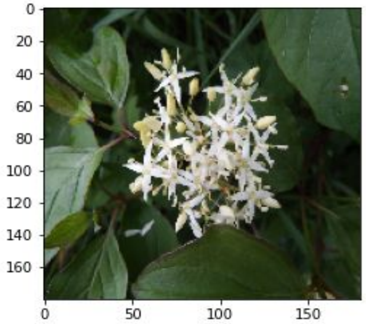
Poison Oak

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



Poison Sumac

This image is 99.99 percent non-poisonous and 0.01 percent poisonous.



Non-Poisonous Plant

Transfer Learning - Workflow

01 Load & split dataset using image_dataset_from_directory() & augment images using Keras' data augmentation layers

Performed an 80/20 split on dataset & used RandomFlip & RandomRotation layers

02 Build, compile & train the model

a) Instantiate a base model and load pre-trained weights into it.
b) Freeze all layers in the base model by setting trainable = False.
c) Create a new model on top of the output of one (or several) layers from the base model.

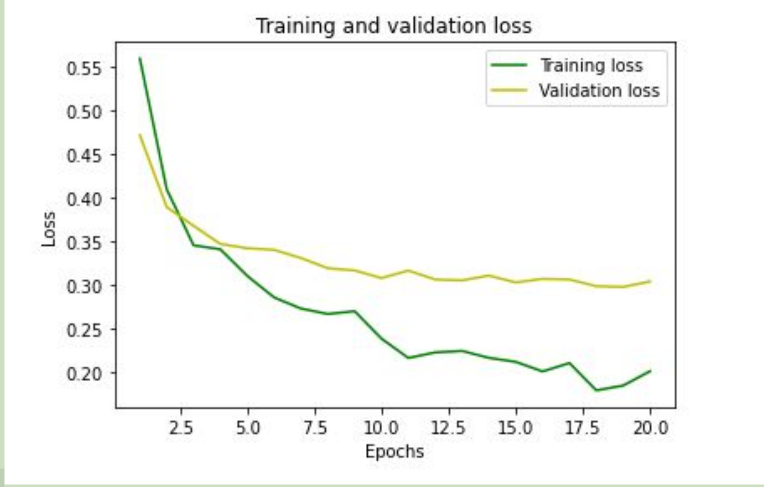
03 Evaluate model performance & use to predict on test set

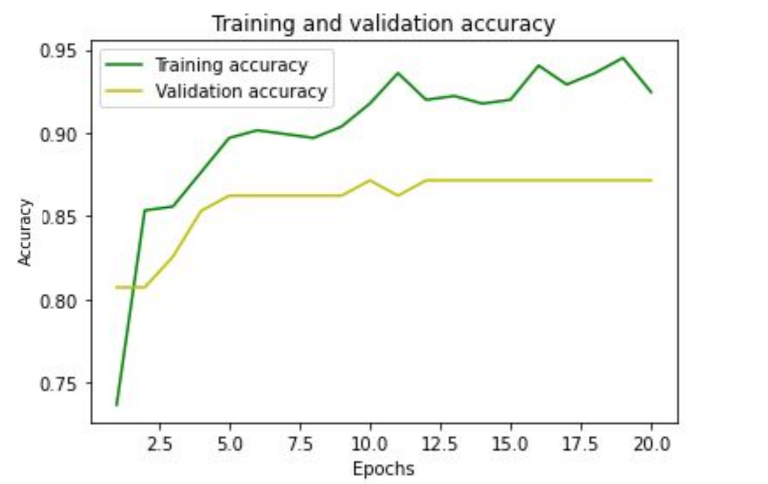
Goal is to maximize validation accuracy

Transfer Learning

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 180, 180, 3)]     0

 sequential (Sequential)     (None, 180, 180, 3)       0

 rescaling (Rescaling)       (None, 180, 180, 3)       0

 xception (Functional)       (None, 6, 6, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 1)                 2049

=================================================================
Total params: 20,863,529
Trainable params: 2,049
Non-trainable params: 20,861,480
_________________________________________________________________
```

Transfer Learning - Training Results





Training Loss = 0.202
Validation Loss = 0.304

Training Accuracy = 92.45%
Validation Accuracy = 87.16%

Validation accuracy is slightly lower than training accuracy

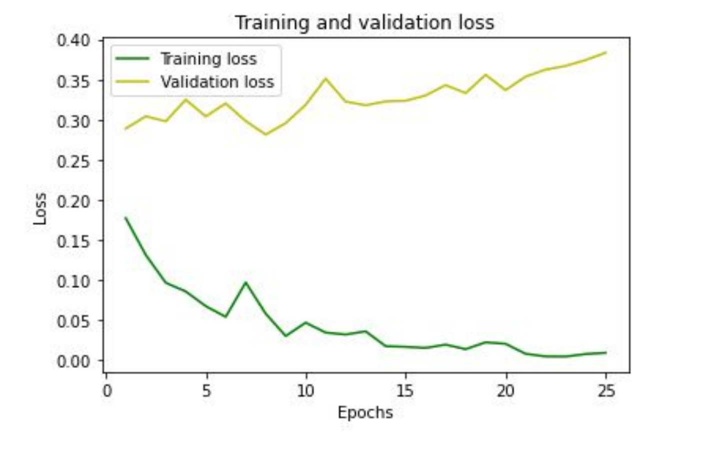
Transfer Learning (Post-Tuning) - Model Summary

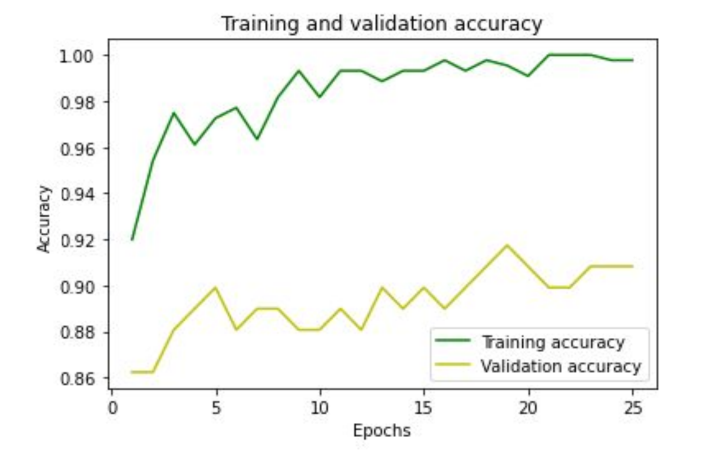
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 180, 180, 3)]     0

 sequential (Sequential)     (None, 180, 180, 3)       0

 rescaling (Rescaling)       (None, 180, 180, 3)       0

 xception (Functional)       (None, 6, 6, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 1)                 2049

=================================================================
Total params: 20,863,529
Trainable params: 20,809,001
Non-trainable params: 54,528
```

# of trainable params increases after we unfreeze the model

Transfer Learning (Post-Tuning) Training Results

Training Accuracy & Validation Accuracy improved slightly



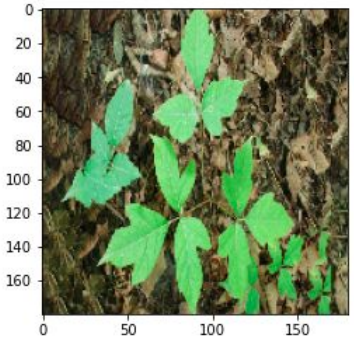


Training Loss = 0.009
Validation Loss = 0.384

Training Accuracy = 99.77%
Validation Accuracy = 90.83%

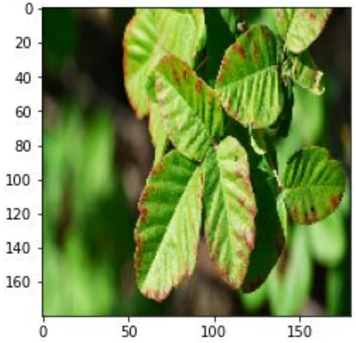
Transfer Learning- Predicting on Test Set

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



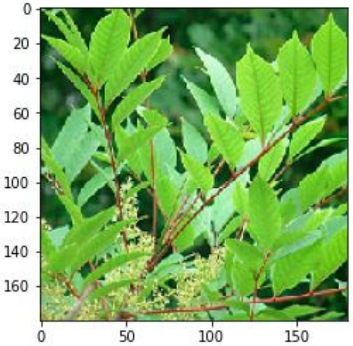
Poison Ivy

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



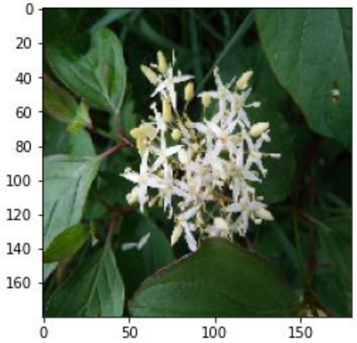
Poison Oak

This image is 0.00 percent non-poisonous and 100.00 percent poisonous.



Poison Sumac

This image is 100.00 percent non-poisonous and 0.00 percent poisonous.



Non-Poisonous Plant

Models Comparison

	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
Simple CNN	77.35%	83.33%	0.485	0.413
Simple CNN (post-tuning)	81.43%	93.33%	0.405	0.202
Xception	95.65%	75.82%	0.114	0.803
Transfer Learning	92.45%	87.16%	0.202	0.304
Transfer Learning (post-tuning)	99.77%	90.83%	0.009	0.384

Key Takeaways

- Images can be corrupted during the download process; learnt how to identify & filter them out
- Identified various ways to apply data augmentation to enhance dataset & their impact on model performance
- Used KerasTuner() to streamline the hyperparameter tuning process
- Further enhance the model, using both images and text data to classify various species of plants.

Thanks for Listening!

Any Questions?

References

- **Poisonous Plants:**
  - https://www.cdc.gov/niosh/topics/plants/default.html
  - https://www.webmd.com/allergies/ss/slideshow-poison-plants
- **Non-poisonous Plants Dataset**
  - https://www.imageclef.org/lifeclef/2015/plant
- **Simple CNN:**
  - https://vijayabhaskar96.medium.com/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720
  - https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/
- **Hyperparameter Tuning using KerasTuner**
  - https://keras.io/keras_tuner/
- **Mini-Xception Model**
  - https://keras.io/examples/vision/image_classification_from_scratch/#two-options-to-preprocess-the-data
- **Transfer Learning**
  - https://keras.io/guides/transfer_learning/