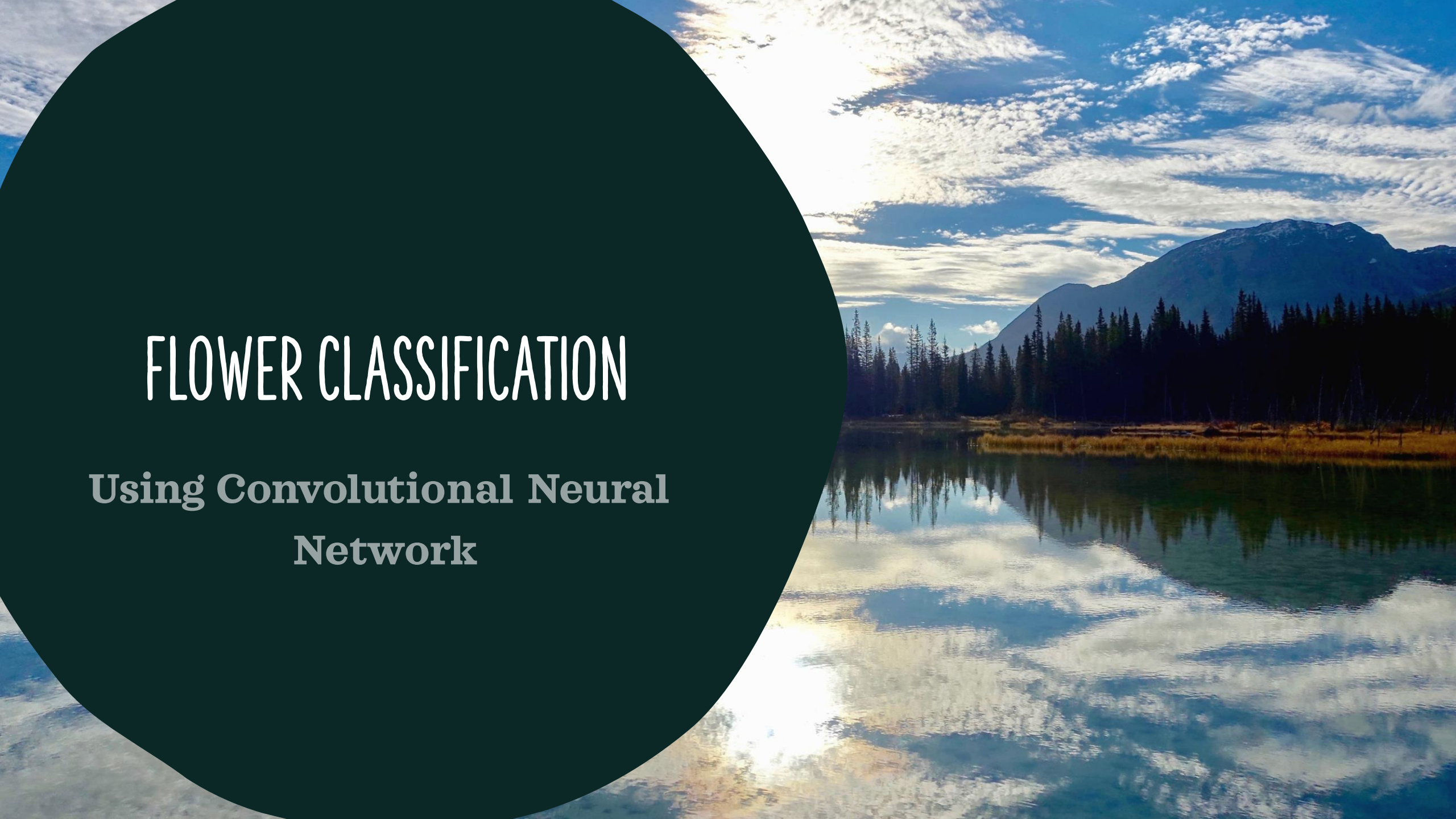


FLOWER CLASSIFICATION

Using Convolutional Neural
Network



WHAT IF YOU ARE SEARCHING FOR A SUNFLOWER

And you were given a dozen of
photograph that is similar to sunflower...

How will you differentiate which one?



Even Taxonomy and experts
find it hard to differentiate the
species of flower...



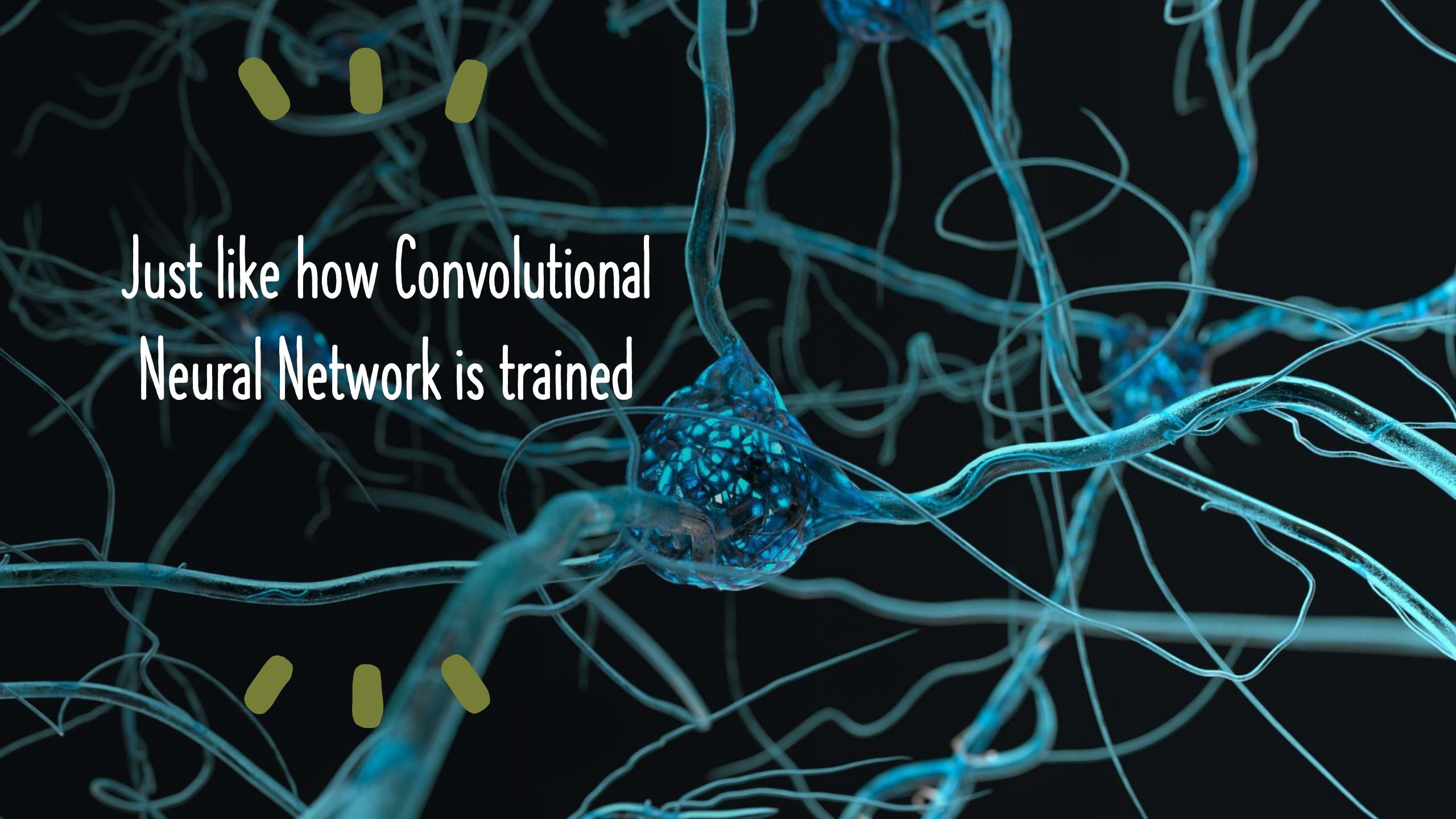
They needed key data

To examine the plant
structure...




And classify the species of the
flower.





Just like how Convolutional
Neural Network is trained



Convolutional Neural Network or ConvNet

Is a class of Artificial Neural Network (ANN) that is mostly applied to identify images by various techniques and algorithm.

([wikipedia.org](https://en.wikipedia.org))

It is inspired by biological process that the human mind perceive when identifying an object based on an image.



Building a Convolutional Neural Network with Flower Classification Model

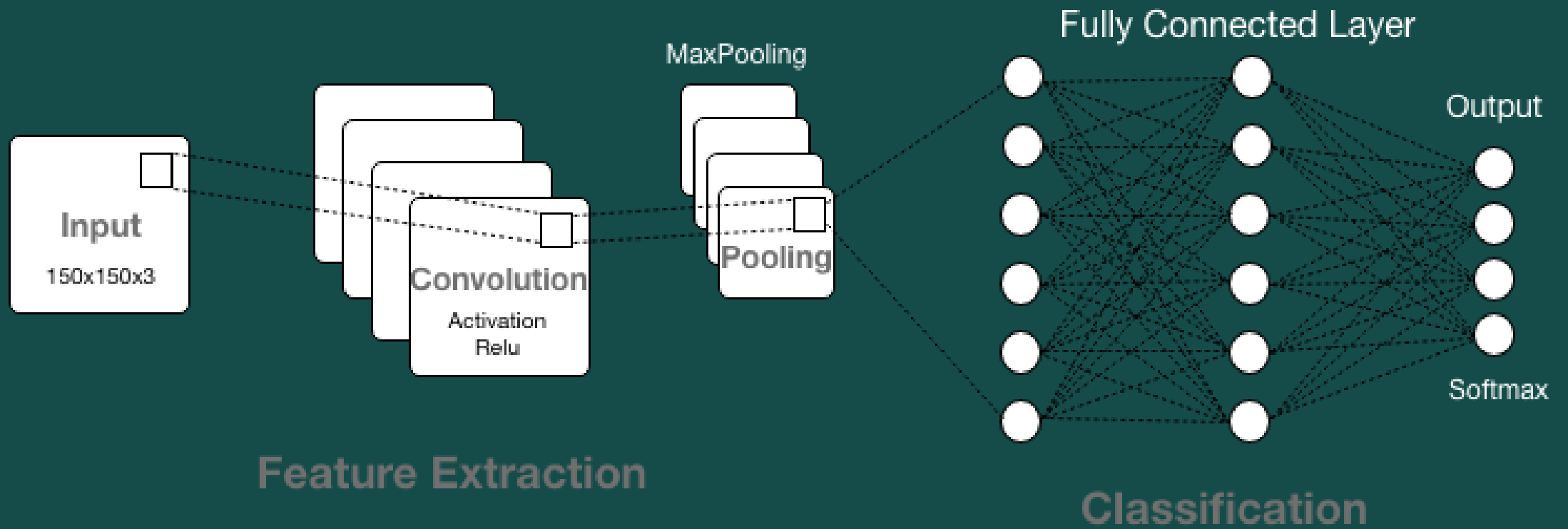


The Dataset

The dataset used in this project is based on Oxford-102 flower classification. I used 20 species of flower from the dataset and label them accordingly. Some images were downloaded from the internet to add on complexity.

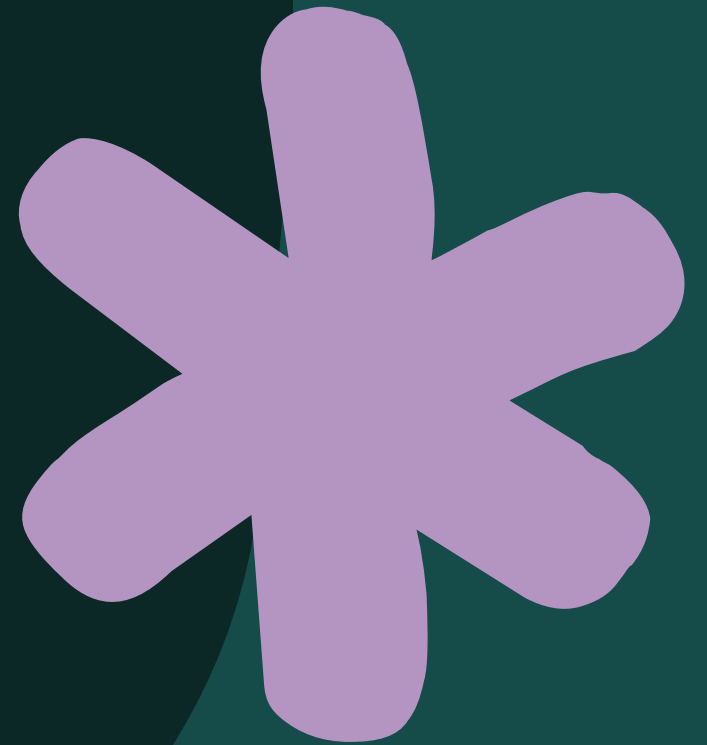


Project Architecture



Input layer

Flower images were transformed into vector of data to form a training dataset for the model. It is uniformed with a dimension of 150x150x3 sizes of images.





Hidden layer

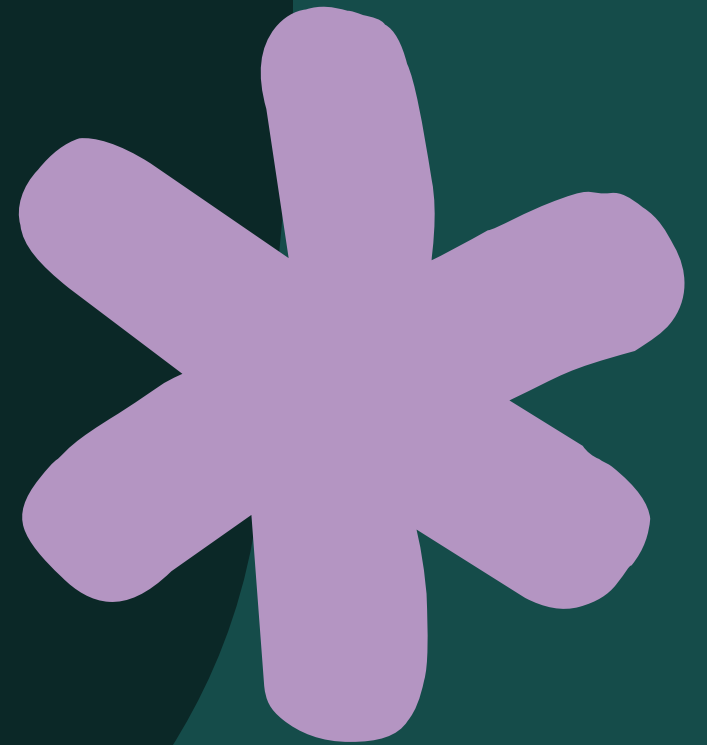
Hidden layer is where the computation of neurons (weights and biases) are proceeding to produce a close to accurate output. It is called hidden because the input and output are obscure in the training dataset. In convolutional neural network, it composes of convolution layer, pooling layer and fully connected layer (Goodfellow, 2017).

There are three layer that this project compose of.

1. Convolution Layer

The feature extraction stage that processes two functions by means of a kernel or filter to produce a new function called feature map (Albawi et al., 2017).

In this phase the filters were set to 32, 64, 96, 128 respectively with kernel size of 3x3, 'same' padding and Rectified Linear Unit (ReLU) as activation function.





2. Pooling Layer

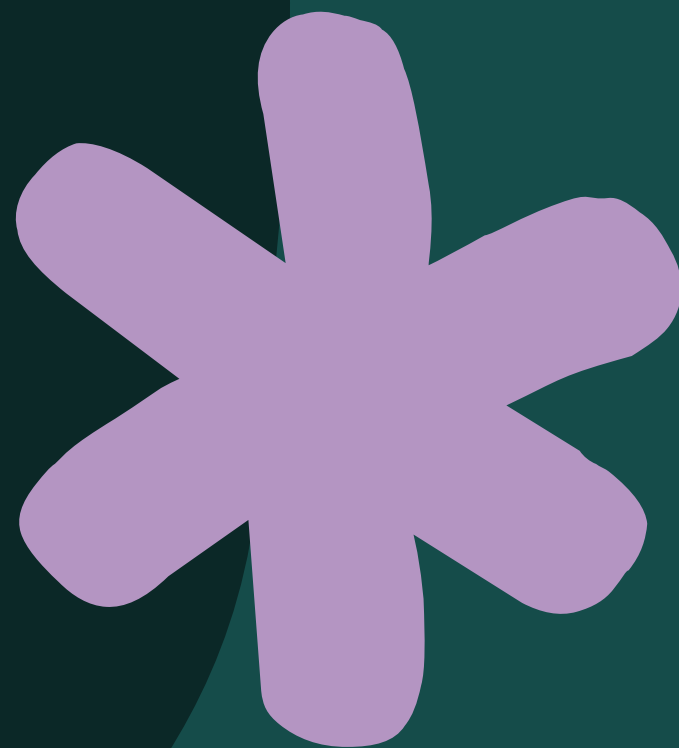
Pooling layer reduces the dimensionality of the feature map to reduce computing time and tackle overfitting. On the process of dimensional reduction, the neurons are being subjected to invariance translation or producing an output layer that emphasizes the important portion of the input layer with minor changes to the principal value (Brownlee, 2019).

In this stage Max pooling is used that acquire the maximum value from the previous layer. The window size is 2×2 with the stride of 2 pixel.

3. Fully Connected Layer

The stage in Convolution Neural Network where the flatten vector from previous layer is transform into a single vector by performing vector transformation which consist of 1 affine function and 1 Non-Linear function (Nikhil, 2017).

Two Fully connected layer were constructed with Relu as the activation function with a dropout of 0.8 to minimize overfitting.





Output layer

This is the final layer of the Neural Network which shows the final output from the previous layer with the computed probability for each label category.

As for this project, the 20 species of flower were used as a label to produce the dense image features with softmax as the activation function.

Data Preparation



Data Normalization

Data normalization is a necessary step to create a similar scale of the input data ranging from 0 to 1 for optimization and to improve learning rate of the model (Brownlee J., 2019).

OpenCV's normalize function is utilized to carry out this process. This function centers the data by subtracting the dataset mean and dividing them by the standard deviation.





One Hot Encoding

One Hot Encoding is done to transform categorical data into numerical value. Further, setup the label for true value to use as a basis for probability classification (Gulli and Pal, 2017).

Sklearn preprocessing LabelEncoder is used to perform the process.

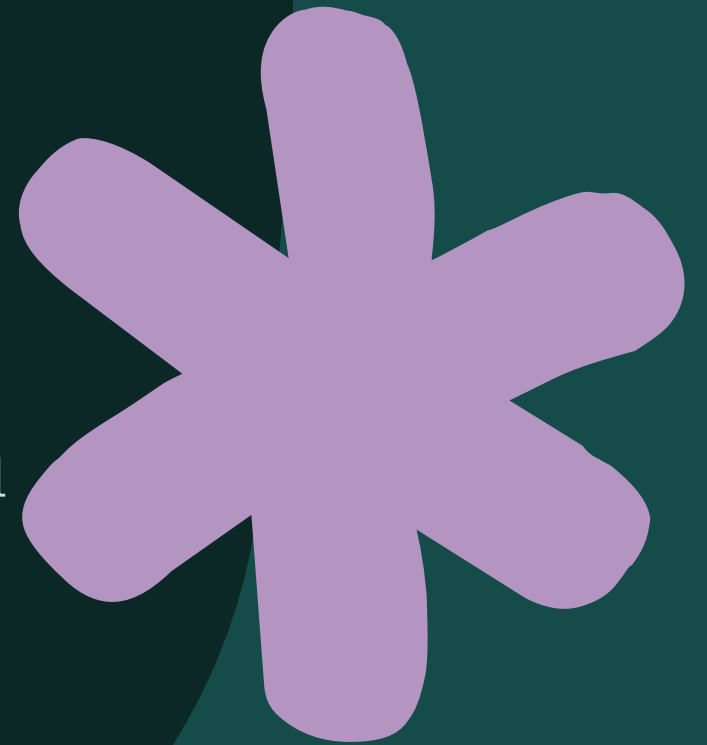
Data Augmentation

Data augmentation can help increase the size of the dataset and prevent overfitting (Kusnetzky, 2011).

Keras library has provided better access for augmenting data. ImageDataGenerator class has attributes to produce different orientation of input images.

The following are used to augment each image:

- random rotation of 0-to-180-degree angle
- enlarge image by 25%
- create shifted images by 10% vertically and horizontally
- finally flipping the image horizontally



Train and Test



Training and testing is a method used to measure the accuracy of the model. The training size for this project is 75% while 25% for testing.



Tuning the
Hyperparameters and training the
model



Hyperparameter tuning is the process of finding the optimal combination of those parameters that minimize cost functions (Gulli and Pal, 2017).

Heuristic tuned Hyperparameters for this task :

- Epochs - [80, 100, 120]
- Batch Size - [50, 60, 64, 70]
- Learning Rate - [0.001, 0.002]
- Dropout - [0.5, 0.6, 0.7, 0.8, 0.9]



I created task for the Hyperparameter tuning that I found most suited to the model based on the previous experiments. The first 5 tuning is test with 70% training set then maximize it to 75% on the last 5 task.

Tasks	Epochs	Batch size	Learning Rate	Drop out (2 FC)
1 (70%)	100	64	0.001	[0.5, 0.8]
2	80	50	0.002	[0.5, 0.8]
3	100	64	0.002	[0.7, 0.8]
4	80	70	0.001	[0.7, 0.8]
5	100	70	0.001	[0.6, 0.8]
6 (75%)	100	60	0.002	[0.6, 0.8]
7	100	64	0.002	[0.7, 0.8]
8	100	60	0.001	[0.8, 0.8]
9	120	50	0.001	[0.7, 0.9]
10	120	60	0.001	[0.7, 0.9]

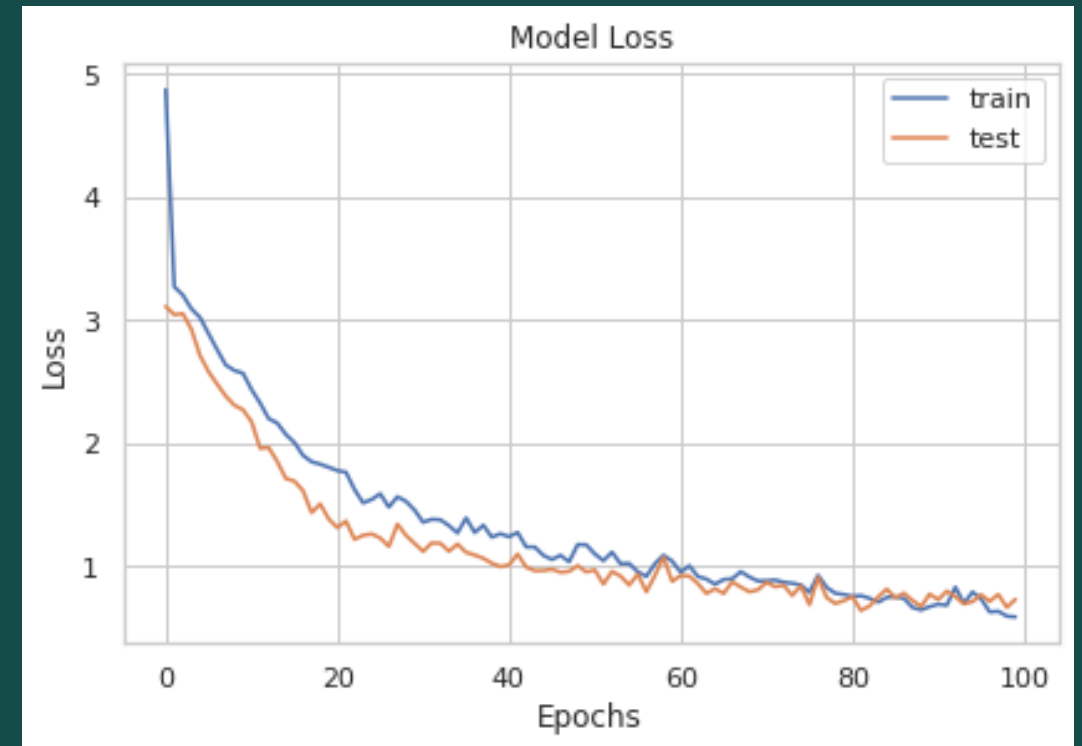
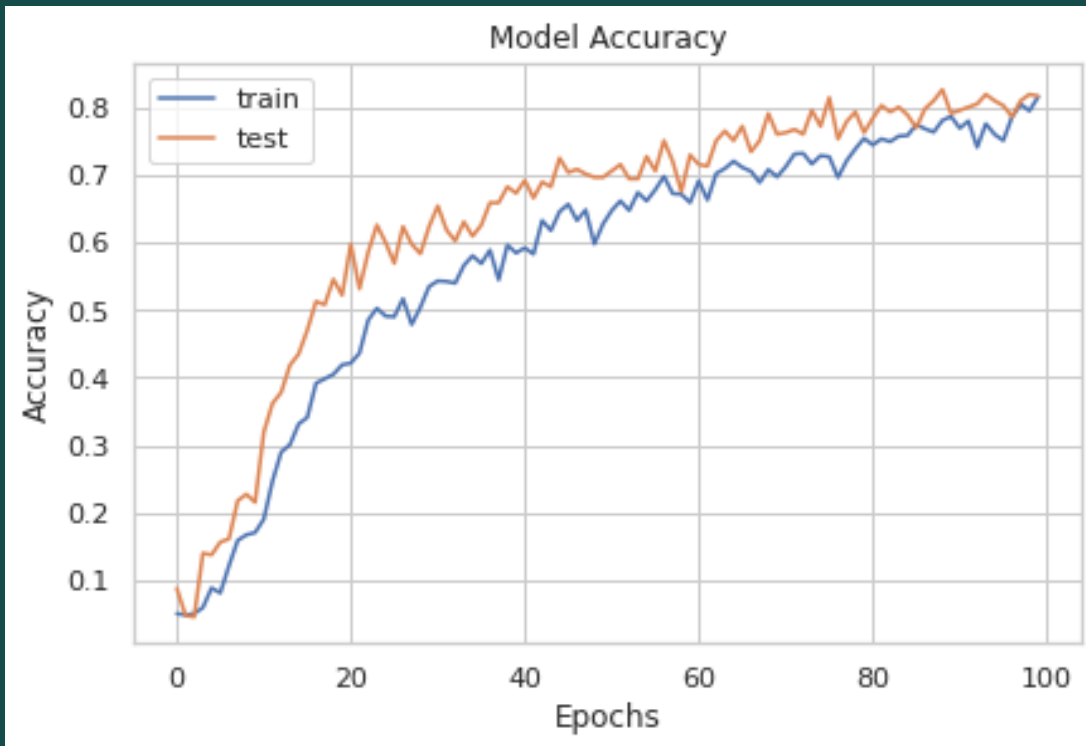
Result



Based on the tuned Hyperparameters, below are the result while training the model.

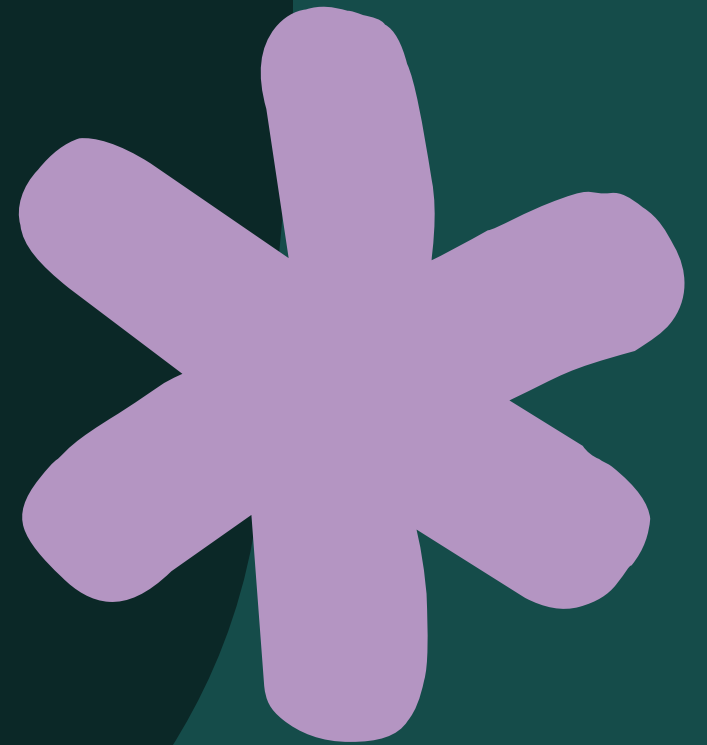
Task	Accuracy	Loss	Val_accuracy	Val_loss
1	0.90	0.28	0.84	0.65
2	0.81	0.57	0.80	0.72
3	0.77	0.73	0.76	0.72
4	0.79	0.68	0.82	0.65
5	0.88	0.38	0.82	0.71
6	0.74	0.86	0.76	0.87
7	0.81	0.61	0.80	0.78
8	0.83	0.49	0.83	0.75
9	0.87	0.42	0.85	0.64
10	0.89	0.38	0.85	0.65

In accordance to the experiment, the best Hyperparameters for the build model that I found that lessen overfitting or underfitting of the data to the model was resulted with 83% of accuracy.



Conclusion

On this study, I created a Convolutional Neural Network that contains 6 hidden layer. Each receive the computed input from the previous layer with activation function of Relu. Data augmentation, optimization, and hyperparameter tuning were conducted to gain a fair result for the model. In this study I encountered a lot of factor that affected the proper training of the model like overfitting and under fitting. This could be possible because of insufficient amount of training images and inability to use Grid search to find the optimal parameters of the model. Further research and study is essential to improve the classification model.



End

