Minor Project - II

Guided by:

Ms. Anupama Arun

Prepared by:

Sukrit Khare(112015144)

Kaustubh Sakhare(112015074)

Om Mandlik(112016019)

Gagan Sharma(112015046)

Automatic Plant Recognition System

Introduction

- The identification of unknown plants requires inherent knowledge of an expert botanist.
- Most successful method to identify plants correctly and easily is a manual-based method based on morphological characteristics.
- However, this process of manual recognition is often laborious and time-consuming.

- Hence, many researchers have conducted studies to support the automatic classification of plants based on their physical characteristics.
- Systems developed so far use varying number of steps to automate the process of automatic classification.
- These steps involve preparing the leaves collected, undertaking some pre-processing, classification of the leaves, populating the database, training for recognition and finally evaluating the results.

Motivation

- The world bears thousands of plant species, many of which have medicinal values, others are close to extinction, and still others that are harmful to man.

- To use and protect endangered plant species, it is crucial to study and classify plants correctly.

- It will help the local population and taxonomists to improve their knowledge on medicinal plants.

Related Work

Reference	Features	Classifier	Accuracy (%)	Dataset	Training	Testing	Species
Wu *et al.* (2007)	Shape, Veins	PNN	90.3	1800	1480	320	32
Du *et al.* (2007)	Shape	kNN, MMCH	93, 90	400	200	200	20
Du *et al.* (2009)	Shape	kNN	92.3	2000+	1000	1000+	20
Backes *et al.* (2009)	Texture	LDA	89.6	2000	1200	800	10
Hossain and Amin (2010)	Shape	PNN	91.4	1200	Ten-fold cross-validation		30
Du *et al.* (2013)	Curvature, Veins	kNN	87.1	2422	1695	727	30
Amin and Khan (2013)	Curvature	kNN	71.5	1600	1120	480	100
Herdiyeni and Wahyuni (2012)	Texture, Colour	PNN	74.5	2448	1938	510	51
Arai *et al.* (2013)	Wavelets	SVM	95.8	120	96	24	8
Hernandez-Serna and Jimenez-Segura (2014)	Shape, Texture	ANN	92.9	1800	1620	180	32
Le *et al.* (2014)	Kernel Descriptor	SVM	98.5 98.3	1905 1312	1585 649	320 663	32 55
Munisami *et al.* (2015)	Shape, Colour	kNN	87.3	640	Leave-one-out cross-validation		32
Chaki *et al.* (2015)	Shape, Texture	NFC	97.6	930	310	620	31
Siravenha and Carvalho (2015)	Shape	ANN	97.5	1865	Ten-fold cross-validation		32
Carranza-Rojas and Mata-Montero (2016)	Curvature, Texture	kNN	87.2	2345	Leave-one-out cross-validation		66

Problem with these techniques

- Model is Data hungry
- Takes lot of time

Hence, we prefer CNN and transfer learning techniques.

Dataset Description

- The Swedish leaf dataset has been captured as part of a joined leaf classifcation project between the Linkoping University and the Swedish Museum of Natural History.
- It contains images of isolated leaf scans on plain background of 10 Swedish tree species, with 75 leaves per species (750 images in total).
- This dataset is considered very challenging due to its high inter-species similarity.

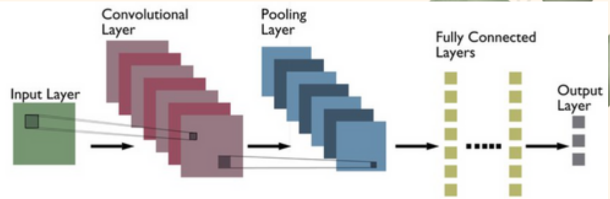
Dataset Sample

METHODOLOGIES

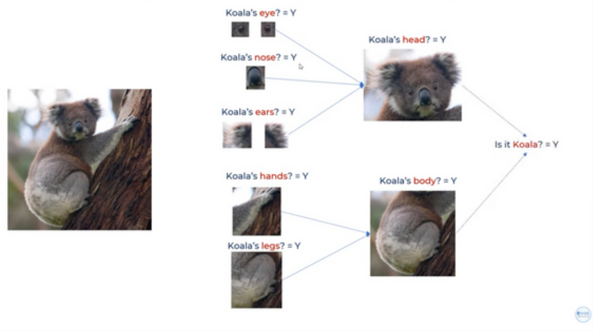
- CNN
- Transfer Learning
  1] Feature Extraction
  2] Fine Tuning
- Transfer Learning Models
  1] DenseNet121
  2] VGG16
  3] Inception ResnetV2
  4] Xception

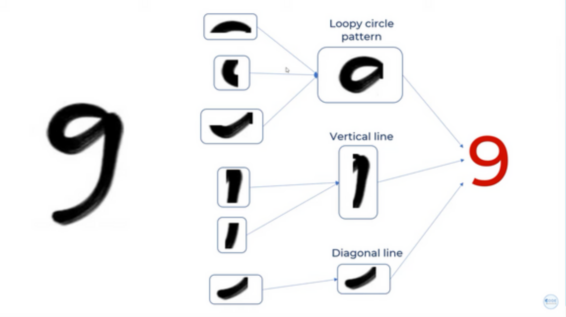
CNN

- A convolutional neural network is a network architecture for deep learning that learns directly from data.
- CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories.



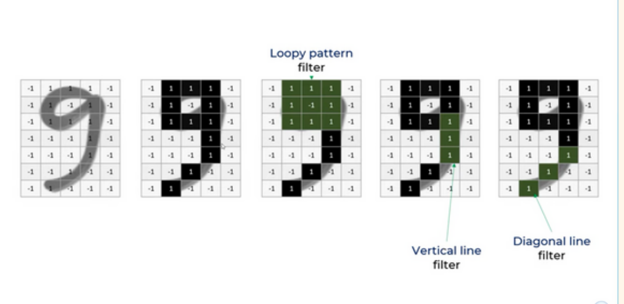
CNN


Koala's eye? = Y
Koala's nose? = Y
Koala's ears? = Y
Koala's head? = Y
Koala's hands? = Y
Koala's legs? = Y
Koala's body? = Y
Is it Koala? = Y



Loopy circle pattern
Vertical line
Diagonal line
9

Image is a 2d matrix of pixels



Convolutional layer

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

1	1	1
1	-1	1
1	1	1

Filter

- Filters are feature detectors
- Filters can be 3d as well

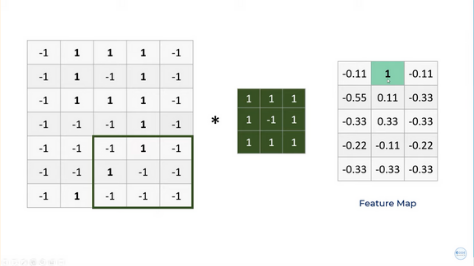
$$-1+1+1-1-1-1-1+1+1 = -1 \rightarrow -1/9 = -0.11$$

-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

*

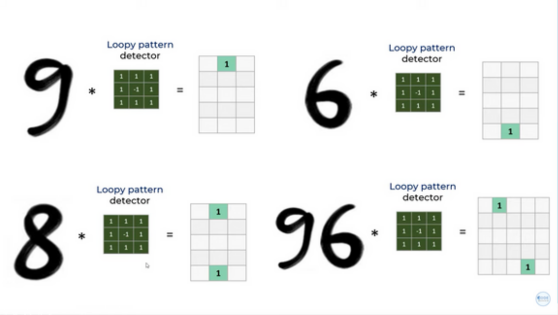
1	1	1
1	-1	1
1	1	1

-0.11		

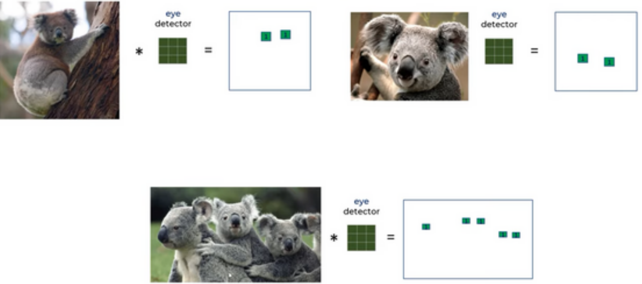


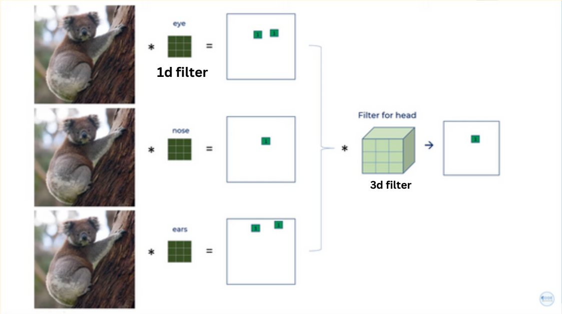
- By doing this convolutional operation we create a feature map

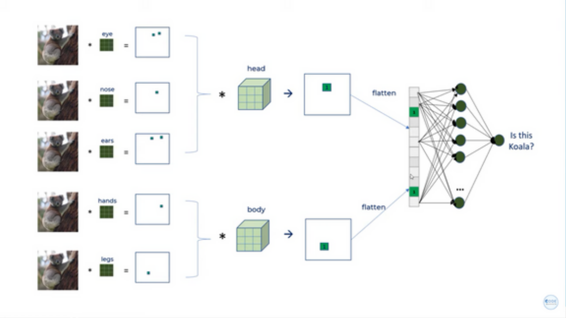
Relu function to bring non linearity in our model
1 or close to 1 value represent feature


9 *
Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
=
1
6 *
Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
=
1
8 *
Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
=
1
1
96 *
Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
=
1
1

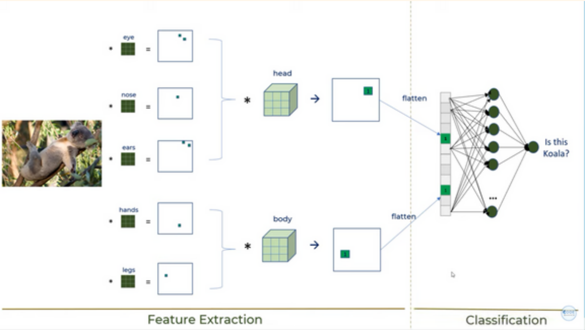
Location invariant: It can detect eyes in any location of the imag




eye
*
=
1d filter
nose
*
=
ears
*
=
Filter for head
*
3d filter

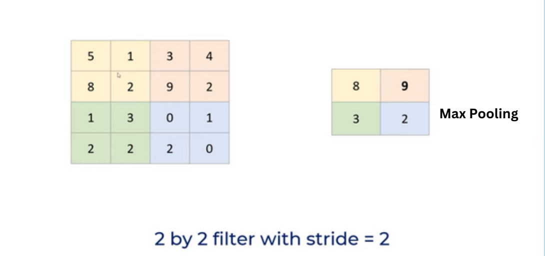


eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this
Koala?

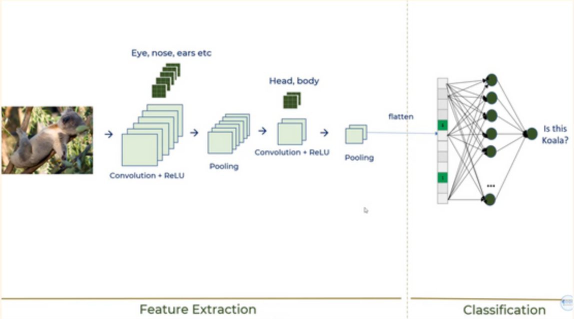


eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this Koala?
Feature Extraction
Classification

Pooling



- Pooling used to reduce the size /dimension
- Computation reduces, overfitting prevented
- Model tolerant towards variations


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this
Koala?
Feature Extraction
Classification

Transfer Learning

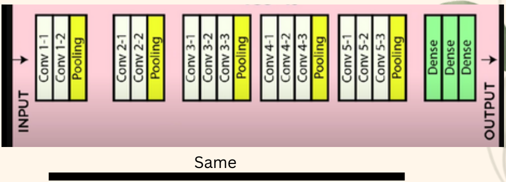
A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task. You either use the pretrained model as it is or use transfer learning to customize this model to a given task.

We can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

Learning Experience

Feature Extraction

Last dense layers replaced by our own own dense layers(classification layers), convolutional base freezed and we retrain the model on our data.



Fine Tuning

- We retrain some of the last convolutional layers , we do not freeze them

Fine Tuning

[19]:

```
history3 =model.fit(train_ds, epochs =15, validation_data = val_ds,callbacks=[WandbCallback()])
run.finish()
```

```
Epoch 1/15
52/52 [==============================] - 113s 2s/step - loss: 1.7959 - accuracy: 0.4212 - val_loss: 0.6765 - val_accuracy: 0.8182
Epoch 2/15
52/52 [==============================] - 88s 2s/step - loss: 0.3983 - accuracy: 0.8865 - val_loss: 0.1642 - val_accuracy: 0.9545
Epoch 3/15
52/52 [==============================] - 85s 2s/step - loss: 0.1523 - accuracy: 0.9596 - val_loss: 0.1129 - val_accuracy: 0.9636
Epoch 4/15
52/52 [==============================] - 89s 2s/step - loss: 0.1467 - accuracy: 0.9500 - val_loss: 0.1613 - val_accuracy: 0.9545
Epoch 5/15
52/52 [==============================] - 88s 2s/step - loss: 0.1569 - accuracy: 0.9404 - val_loss: 0.1121 - val_accuracy: 0.9636
Epoch 6/15
52/52 [==============================] - 92s 2s/step - loss: 0.0832 - accuracy: 0.9731 - val_loss: 0.0431 - val_accuracy: 1.0000
Epoch 7/15
52/52 [==============================] - 87s 2s/step - loss: 0.0615 - accuracy: 0.9788 - val_loss: 0.0232 - val_accuracy: 0.9909
Epoch 8/15
52/52 [==============================] - 86s 2s/step - loss: 0.0681 - accuracy: 0.9808 - val_loss: 0.0846 - val_accuracy: 0.9636
Epoch 9/15
52/52 [==============================] - 87s 2s/step - loss: 0.0612 - accuracy: 0.9827 - val_loss: 0.0164 - val_accuracy: 1.0000
Epoch 10/15
52/52 [==============================] - 85s 2s/step - loss: 0.0342 - accuracy: 0.9904 - val_loss: 0.0204 - val_accuracy: 0.9909
Epoch 11/15
52/52 [==============================] - 86s 2s/step - loss: 0.0256 - accuracy: 0.9904 - val_loss: 0.0753 - val_accuracy: 0.9727
Epoch 12/15
52/52 [==============================] - 86s 2s/step - loss: 0.0363 - accuracy: 0.9942 - val_loss: 0.0252 - val_accuracy: 0.9909
Epoch 13/15
52/52 [==============================] - 87s 2s/step - loss: 0.0170 - accuracy: 0.9904 - val_loss: 0.0144 - val_accuracy: 1.0000
Epoch 14/15
52/52 [==============================] - 86s 2s/step - loss: 0.0419 - accuracy: 0.9885 - val_loss: 0.1967 - val_accuracy: 0.9636
Epoch 15/15
52/52 [==============================] - 86s 2s/step - loss: 0.0273 - accuracy: 0.9942 - val_loss: 0.0095 - val_accuracy: 1.0000
```

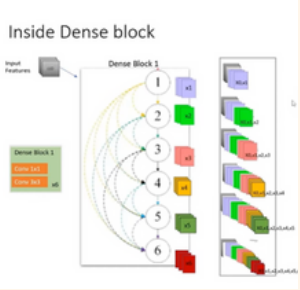
Waiting for W&B process to finish... (success).

Data Augmentation

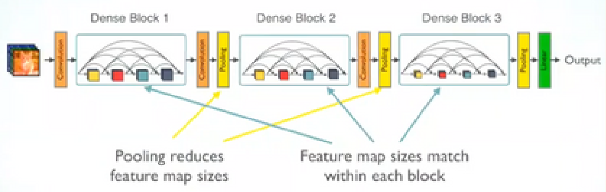
- New samples
- Make model robust preventing overfitting
- Generalization done
- Variety introduced

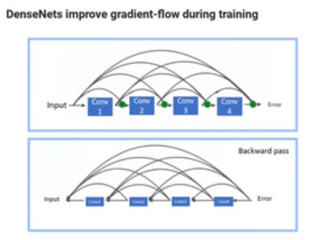
```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=30,
                                   zoom_range=0.15,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.15,
                                   horizontal_flip=True,
                                   fill_mode="nearest")

test_datagen=ImageDataGenerator(rescale=1./255)
```

DenseNet121



- In densenet each layer receives feature map from previous layer and add some features on top of existing feature map.
- But we concatenate this feature map only if the size of those feature map is same.
- So to overcome this drawback we send info from dense layer to transition layer. So transition layer down sample the feature map which it receives .
- Densenet collect all kind of feature from each layer


Dense Block 1
Dense Block 2
Dense Block 3
Convolution
Convolution
Pooling
Convolution
Pooling
Pooling
Linear
Output
Pooling reduces feature map sizes
Feature map sizes match within each block



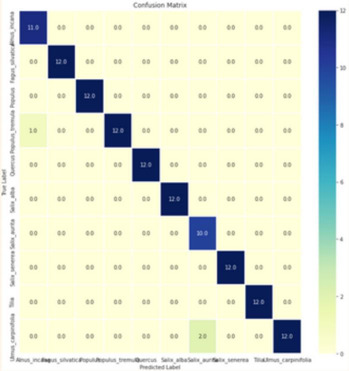
Error signal can also be send earlier in densenet as in backward pass also each layer connected with every other

Advantages of DENSENET

- Strengthen feature propagation.
- Encourage feature reuse .
- Reduce number of parmeters.

val_accuracy
jobType: train
1
0.95
0.9
0.85
Step
0 2 4 6 8 10 12 14
loss
jobType: train
1.5
1
0.5
0
Step
0 2 4 6 8 10 12 14
epoch
jobType: train
14
12
10
8
6
4
2
0
Step
0 2 4 6 8 10 12 14
accuracy
jobType: train
0.8
0.6
0.4
0.2
0
Step
0 2 4 6 8 10 12 14
val_loss
jobType: train
1
0.8
0.6
0.4
0.2
0
Step
0 2 4 6 8 10 12 14

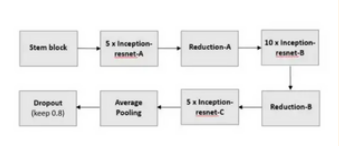

Prediction:Populus
Prediction:Populus
Prediction:Tilia
Truth: Tilia
Prediction:Tilia
Truth: Salix_alba
Prediction:Salix_alba
Truth: Tilia
Prediction:Tilia
Truth: Salix_alba
Prediction:Salix_alba
Truth: Salix_senerea
Prediction:Salix_senerea
Truth: Fagus_silvatica
Prediction:Fagus_silvatic


Confusion Matrix
True Label
Predicted Label
Alnus_incana
Fagus_silvatica
Populus
Populus_tremula
Quercus
Salix_alba
Salix_aurita
Salix_senerea
Tilia
Ulmus_carpinifolia
0
2
4
6
8
10
12

Inception ResnetV2

- Inception-ResNet-v2 is a convolutional neural architecture that builds on the Inception family of architectures but incorporates residual connections (replacing the filter concatenation stage of the Inception architecture).

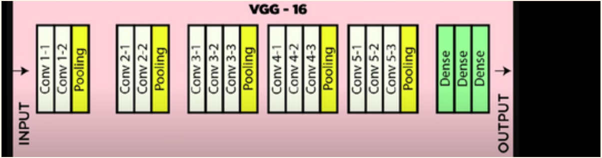
- In the Inception-Resnet block, multiple sized convolutional filters are combined with residual connections. The usage of residual connections not only avoids the degradation problem caused by deep structures but also reduces the training time. The figure shows the basic network architecture of Inception-Resnet-v2.

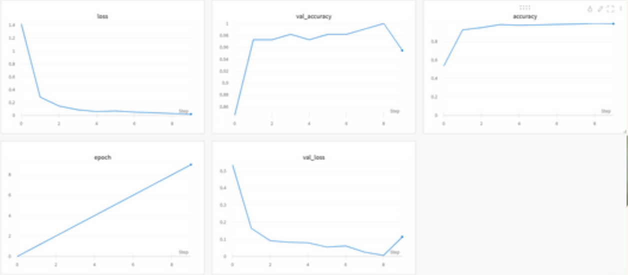



val_accuracy
group: InceptionResNetV2, jobType: train
Step
val_loss
group: InceptionResNetV2, jobType: train
Step
accuracy
group: InceptionResNetV2, jobType: train
Step
loss
group: InceptionResNetV2, jobType: train
Step
epoch
group: InceptionResNetV2, jobType: train
Step

VGG16

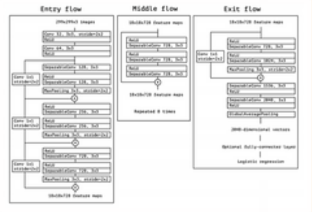
2 Parts-Convolutional Part and Fully connected layer

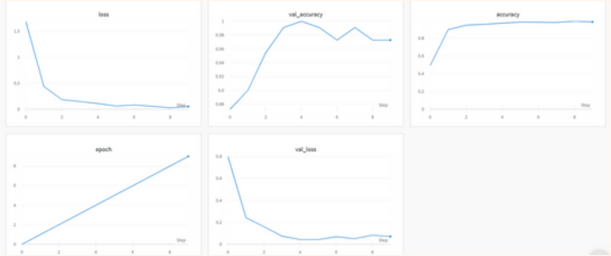



loss
Step
val_accuracy
Step
accuracy
Step
epoch
Step
val_loss
Step

Xception

It is deep convolutional network that consist of depthwise seperable convolution. Xception means extreme inception & it consist of 36 cnn layer which are used to extract features. It is extension of inception model which replaces inception model with depthwise seperable convolution.




loss
val_accuracy
accuracy
epoch
val_loss
Step

Result


val_accuracy
group: Xception, jobType: train
group: DenseNet121, jobType: train
group: InceptionResNetV2, jobType: train
Step
accuracy
group: Xception, jobType: train
group: DenseNet121, jobType: train
group: InceptionResNetV2, jobType: train
Step
loss
group: Xception, jobType: train
group: DenseNet121, jobType: train
group: InceptionResNetV2, jobType: train
Step
val_loss
group: Xception, jobType: train
group: DenseNet121, jobType: train
group: InceptionResNetV2, jobType: train
Step
epoch
group: Xception, jobType: train
group: DenseNet121, jobType: train
group: InceptionResNetV2, jobType: train
Step

MODEL	ACCURACY
DENSENET -	99.72
XCEPTION-	99.16
VGG16-	98.33
INCEPTION RESNETV2-	97.50

Conclusion

Since DenseNet has the highest accuracy,
It will be used for identification of leaves in our model

Research Gap & Future Scope

- Cultivar recognition- in which the leaf image patterns usually have very subtle difference among cultivars has not yet received considerable attention in computer vision community.

- Expand the plant recognition system by adding more dataset.

-

- Researches in this area can make possible followings as Identification of poisonous plants, to improve yields, Protecting Crops, Better Plantation and caring of Plants.

- It can provide more details about plants such as:- Plant form or shape, Plant size and age, Where these plants grow such as Is the plant growing in wet or dry conditions, or in a sunny or shady area.

Reference

- https://d1wqtxts1xzle7.cloudfront.net/83067799/Paper_24-Automatic_Recognition_of_Medicinal_Plants-with-cover-page-v2.pdf?Expires=1665082165&Signature=Kz1RB7n4Z1zG4xGWqoAnCV6QTv2Yov80dZgzUnaLjr77sDDALvM4LyN9Ovm9Z42VgCwbSOuVHLtF~dUmCm1abEXY4k-G2lvd7qAlfokwXf9qtQzf~B-kKQOO4j-ipReGO3Wl4aAFDWycBiBgB74hynltDk69L2lXa1mlhH8bSn1I4Knv2k7O~x-wbL6~WUWxFcGmNgUDDSYZ0IxHUXVphaec0olM7zbMVsDbFSkB-jV1wne3jEg1lUg8-lEkpVOTiYUCuYvUiIgMNs0fcqfpYj6ftc0stwLTKj5zvo18AhcZ1AHCTtuRmvImVNJUQhOgcWPhMFrVouFpvN9I-cjT-Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

- https://ictactjournals.in/paper/IJIVP_Vol_9_Iss_2_Paper_10_1919_1925.pdf

- https://doi.org/10.1007/s11831-016-9206-z

- https://ceur-ws.org/Vol-1391/121-CR.pdf

- http://www.cvl.isy.liu.se/en/research/datasets/swedish-leaf/.


Thank you