CLASSIFICATION OF PLANT SEEDLINGS FROM WEEDS

A PROJECT REPORT

Submitted by

PRAHADEESH K - 2019115**066**
RITHISH A - 2019115**079**
ROHITH M S R - 2019115**081**

Submitted for the

Data Analytics and Cloud Computing Laboratory Project

TABLE OF CONTENTS

1. INTRODUCTION

Plants which are intentionally grown in fields and gardens are called cultivated plants. All other plants which are not supposed to be there and grow unwanted, are called weeds. A weed is therefore a plant growing "in the wrong place".

Weed control is vital to agriculture, because weeds decrease yields, increase production costs, interfere with harvest, and lower product quality. Weeds also impede irrigation water-flow, interfere with pesticide application, and harbour disease organisms.

1.1 PROBLEM STATEMENT:

Weed management has a vital role in applications of agriculture domain. One of the key tasks is to identify the weeds after few days of plant germination which helps the farmers to perform early-stage weed management to reduce the contrary impacts on crop growth.

Thus, we aim to classify the seedlings of crop and weed species. In this work, we propose a plant seedlings classification using the benchmark plant seedlings dataset. The dataset contains the images of 12 different species where three belongs to plant species and the other nine belongs to weed species.

1.2 OBJECTIVES:

- Distinguish between the different weed seeding and crop seeding of 12 different plant species
- Take an image of a weed or a crop seedling and output the correspondent species and its type from our 12 classes.

2. LITERATURE SURVEY

2.1 CONVOLUTIONALNEURAL NETWORK (CNN):

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. It is a deep learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

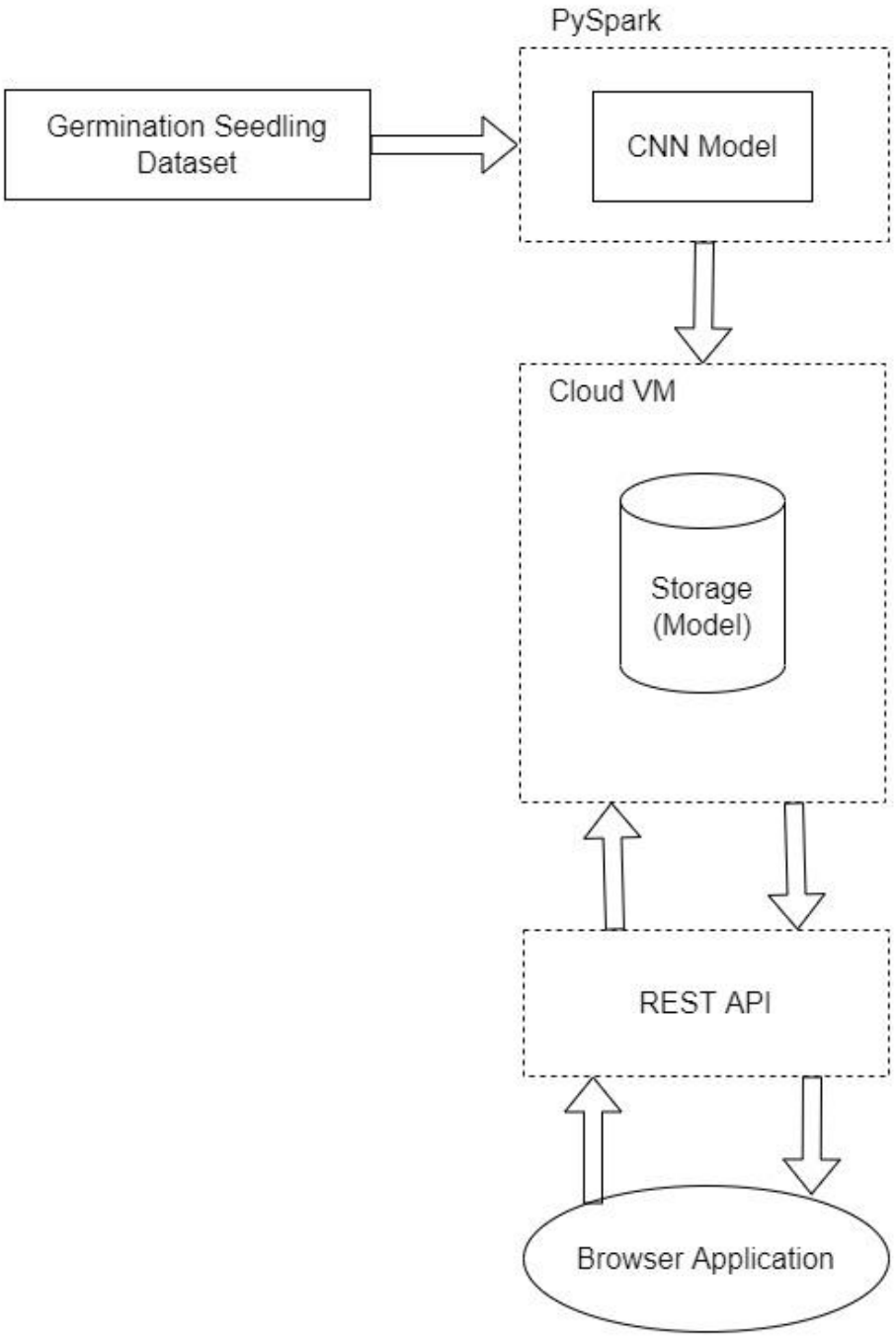
The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

There are 5 Steps for image classification by CNN.

- Image Data Loading (+EDA)
- Image Data Preprocessing (Label Encoding, Data Augmentation, etc..)
- Modelling (Building, Fitting,etc..)
- Prediction of Test Dataset

3. SYSTEM DESIGN

3.1 OVERALL DESIGN:



This dataset contains images of crop and weed seedlings. The images are grouped into 12 classes.These classes represent common plant species in Danish agriculture. Each class contains rgb images that show plants at

different growth stages. The images are in various sizes and are in png format.

3.2 DATA AUGMENTATION:

Data Augmentation is a technique used to expand or enlarge your dataset by using the existing data of the dataset. We apply different techniques to expand our dataset so that it will help to train our model better with a large dataset. If you are having a small dataset and if you use that dataset to train your model and overfit the data. So to increase the ability and performance of your model, or to generalize our model we need a proper dataset so that we can train our model. Data Augmentation helps you to achieve this. Image Augmentation is one of the techniques we can apply on an image dataset to expand our dataset so that we can prevent overfitting and our model generalizes well.

Image Augmentation Techniques:

- Rotation
- Width Shifting
- Height Shifting
- Brightness
- Shear Intensity
- Zoom
- Channel Shift
- Horizontal Flip
- Vertical Flip

4. IMPLEMENTATION

4.1 TOOLS AND LIBRARIES USED:

- Tensorflow and Keras - For adding the layers such as dropout, dense, max pooling, normalization in the CNN
- boto3 - the AWS SDK for Python (Boto3) to create, configure, and manage AWS services, such as Amazon S3.
- cv2 - For processing the image such as cleaning and removing the background.
- Flask - For the web application for the users

4.2 CNN MODEL:

In general, neural networks accept a single vector as input, transform it to a series of hidden layers, which in turn is made up of set of neurons that are fully connected to all neurons in the previous layer. Neurons of the same layer are independent and do not share any connections.

After the hidden layers, is the last fully connected layer which is also called the ‘output layer’, where each node outputs score for each class. The downside of regular neural network is that they don’t scale well to full images. It's mainly because with images of decent size, the number of neurons and weights that the network must accommodate becomes unmanageable. This is where Convolutional Neural Network comes to rescue with its neurons arranged in 3 dimensions (width, height, depth). Each of the layer in CNN accepts 3D input volume and transforms it into 3D output volume.

CNN algorithm consists of several convolution (CNV) operations followed of the image sequentially which is followed by pooling operation (PL) to generate the neurons feed into fully connected (FC) layer.

Input: 2D image data with HSV (hue saturation value)

The first is the **convolutional (Conv2D) layer**: It is like a set of learnable filters. Each filter transforms a part of the image (defined by the

kernel size) using the kernel filter. The kernel filter matrix is applied on the whole image. Filters can be seen as a transformation of the image. The CNN can isolate features that are useful everywhere from these transformed images (feature maps).

The second important layer in CNN is the **pooling (MaxPool2D) layer**: This layer simply acts as a downsampling filter. It looks at the 2 neighboring pixels and picks the maximal value. These are used to reduce computational cost, and to some extent also reduce overfitting. We have to choose the pooling size (i.e the area size pooled each time) more the pooling dimension is high, more the downsampling is important. Combining convolutional and pooling layers, CNN are able to combine local features and learn more global features of the image.

Dropout is a regularization method, where a proportion of nodes in the layer are randomly ignored (setting their weights to zero) for each training sample. This drops randomly a proportion of the network and forces the network to learn features in a distributed way. This technique also improves generalization and reduces the overfitting.

Relu Layer is the abbreviation of Rectified Linear Units. This layer applies the non-saturating activation function. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Flatten layer is use to convert the final feature maps into a one single 1D vector. This flattening step is needed so that you can make use of fully connected layers after some convolutional/maxpool layers. It combines all the found local features of the previous convolutional layers. In the end i used the features in two fully-connected (Dense) layers which is just artificial neural networks (ANN) classifier.

In the last layer (Dense(10,activation="softmax")) the net outputs distribution of probability of each class.

Classification Layer (CL):

This is the final layer of the CNN that converts the output of FC to probability of each object being in a certain class. Typically soft-max type of algorithms are used in this layer.

The model has the following architecture:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 252, 252, 64)      4864
_________________________________________________________________
batch_normalization (BatchNo (None, 252, 252, 64)      256
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 248, 248, 64)      102464
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 124, 124, 64)      0
_________________________________________________________________
batch_normalization_1 (Batch (None, 124, 124, 64)      256
_________________________________________________________________
dropout (Dropout)            (None, 124, 124, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 120, 120, 128)     204928
_________________________________________________________________
batch_normalization_2 (Batch (None, 120, 120, 128)     512
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 116, 116, 128)     409728
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 58, 58, 128)       0
_________________________________________________________________
batch_normalization_3 (Batch (None, 58, 58, 128)       512
_________________________________________________________________
dropout_1 (Dropout)          (None, 58, 58, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 54, 54, 256)       819456
_________________________________________________________________
batch_normalization_4 (Batch (None, 54, 54, 256)       1024
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 50, 50, 256)       1638656
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 256)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 25, 25, 256)       1024
_________________________________________________________________
dropout_2 (Dropout)          (None, 25, 25, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 160000)            0
_________________________________________________________________
dense (Dense)                (None, 256)               40960256
_________________________________________________________________
batch_normalization_6 (Batch (None, 256)               1024
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
batch_normalization_7 (Batch (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 12)                3084
=================================================================
Total params: 44,214,860
Trainable params: 44,212,044
Non-trainable params: 2,816
_________________________________________________________________
```

5. TESTING AND RESULTS

5.1 TESTING:

In [88]:
```
pred_class = np.argmax(predictions, axis=1)
actual_class = np.argmax(y_test, axis=1)

pred_str = le.classes_[pred_class]
actual_str = le.classes_[actual_class]

final_pred = {
    'actual': actual_str,
    'predicted': pred_str
}

final_pred = pd.DataFrame(final_pred)
```

In [89]:
```
final_pred.sample(10)
```

Out[89]:

	actual	predicted
798	Fat Hen	Fat Hen
543	Shepherds Purse	Shepherds Purse
754	Maize	Maize
291	Scentless Mayweed	Scentless Mayweed
357	Small-flowered Cranesbill	Small-flowered Cranesbill
709	Fat Hen	Fat Hen
947	Shepherds Purse	Shepherds Purse
183	Small-flowered Cranesbill	Small-flowered Cranesbill
576	Loose Silky-bent	Loose Silky-bent
318	Fat Hen	Fat Hen

5.2 PERFORMANCE ANALYSIS:

- **Accuracy** is a common metric for binary classifiers; it takes into account both the true positives and true negatives with equal weight.
- Accuracy = TP + TN / dataset-size

```
print('Validation score:', score,'\nValidation accuracy:', acc)
```

```
Validation score: 0.7110856175422668
Validation accuracy: 0.7568420767784119
```

- Precision - Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.

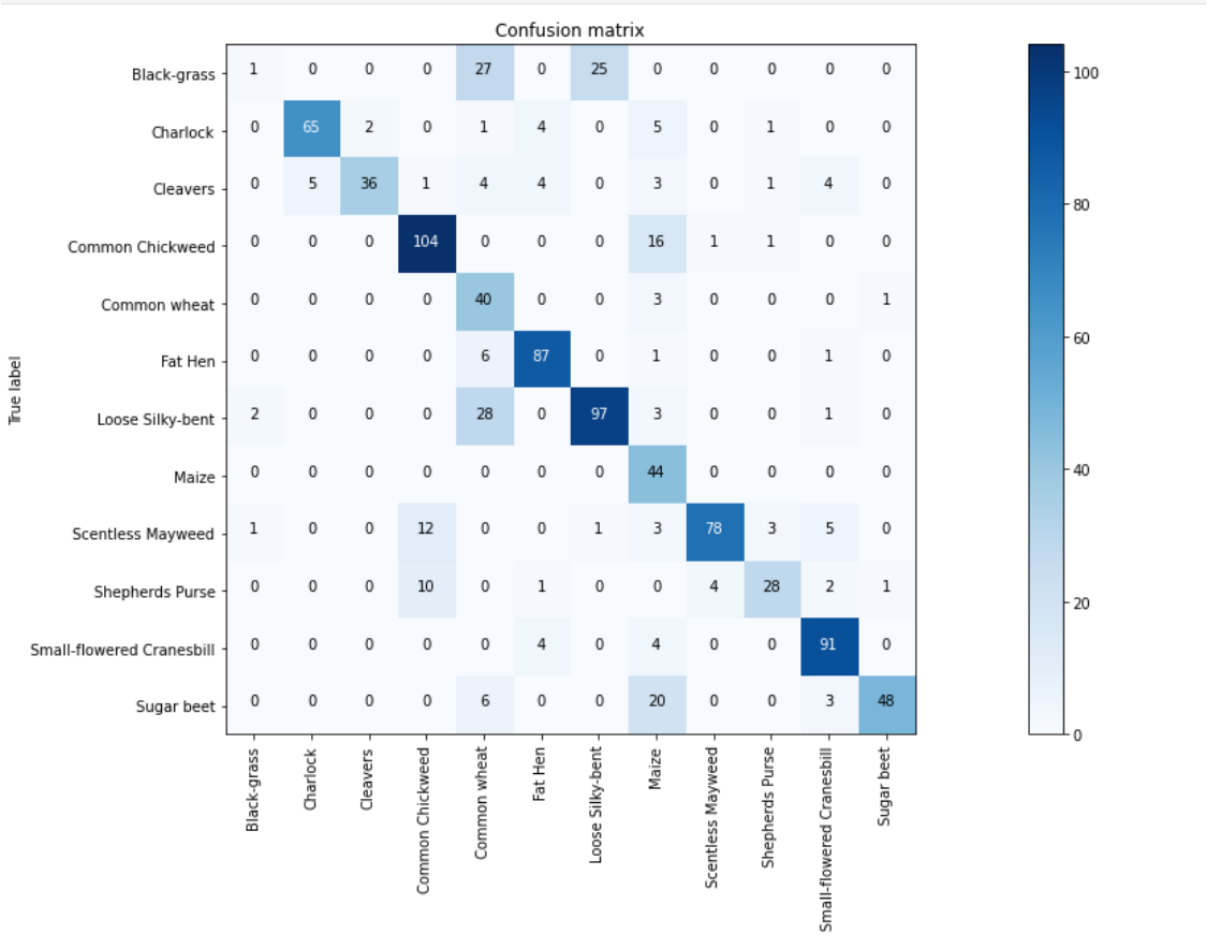
- Recall - Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive).

```
Classification Report:
```

	precision	recall	f1-score	support
Black-grass	0.25	0.02	0.04	53
Charlock	0.93	0.83	0.88	78
Cleavers	0.95	0.62	0.75	58
Common Chickweed	0.82	0.85	0.84	122
Common wheat	0.36	0.91	0.51	44
Fat Hen	0.87	0.92	0.89	95
Loose Silky-bent	0.79	0.74	0.76	131
Maize	0.43	1.00	0.60	44
Scentless Mayweed	0.94	0.76	0.84	103
Shepherds Purse	0.82	0.61	0.70	46
Small-flowered Cranesbill	0.85	0.92	0.88	99
Sugar beet	0.96	0.62	0.76	77
accuracy			0.76	950
macro avg	0.75	0.73	0.70	950
weighted avg	0.79	0.76	0.75	950

- **Confusion Matrix** can be very helpful to see the model drawbacks

Here is the plot for this model:



6. CONCLUSION AND FUTURE WORK

Our intention was to differentiate weeds seedlings from crops seedlings based on their images. We started from different data preprocessing methods which includes data augmentation, image segmentation, and label conversion for a highly usable training set. Then we explored CNN which helped to extract features from the input images. Features are then fed into a logistic regression classifier and a neural network classifier for comparison purpose.

In future, a single image containing multiple plant seedlings as well as video inputs can be given to the model for real-time classification of the seedlings.