

In20-S2-CS5613 - Neural Networks

Group Assignment

Plant Seedlings Images Classification using Convolutional Neural Networks (CNNs)

Group : Avengers

Members : D.P. Jayasuriya (209337M), T.P.N. Silva (209381P), H.T.C Leelananda (209351B)

GitRepo : <https://github.com/DulanGit/Plant-Seedling-Classification>

INTRODUCTION

The objective of this assignment is developing a deep CNN model to classify the images of plant seedlings. This task was carried out with the segmented image with background subtraction of green color.

System Configuration

- Tensorflow version 2.2
- Python 3.6.9
- TensorFlow GPU version 2.2.0
- Ubuntu 20.04
- Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz
- Nvidia GeForce GTX 1650 (4GB)

Data Set

The Data Set used for plant seedling classification consisted of a training data set and a testing data set which contained images of plant seedlings at various growth stages. There were 12 plant species used in the classification.

Data Preprocessing

Cleaning and preprocessing of data is a crucial step of the project. It consisted of 3 phases.

- a. Convert RGB images to HSV format.
- b. Blur the images to remove the noise.
- c. Create a mask to remove the background.
- d. Normalize the pixel values

The images were converted to HSV. HSV color space consists of 3 matrices, 'hue', 'saturation' and 'value'. In OpenCV, value range for 'hue', 'saturation' and 'value' are respectively 0-179, 0-255 and 0-255. 'Hue' represents the color, 'saturation' represents the amount to which that respective color is mixed with white and 'value' represents the amount to which that respective color is mixed with black.

Then the images were filtered using GaussianBlur technique.

It was followed by creating a mask to separate the background and extract the seedling.

```
def preprocess_image(self, image):
    image = cv.resize(image, Preprocess.input_shape)
    # First Use gaussian blur
    blurImg = cv.GaussianBlur(image, (5, 5), 0)

    # Convert to HSV image
    hsvImg = cv.cvtColor(blurImg, cv.COLOR_BGR2HSV)

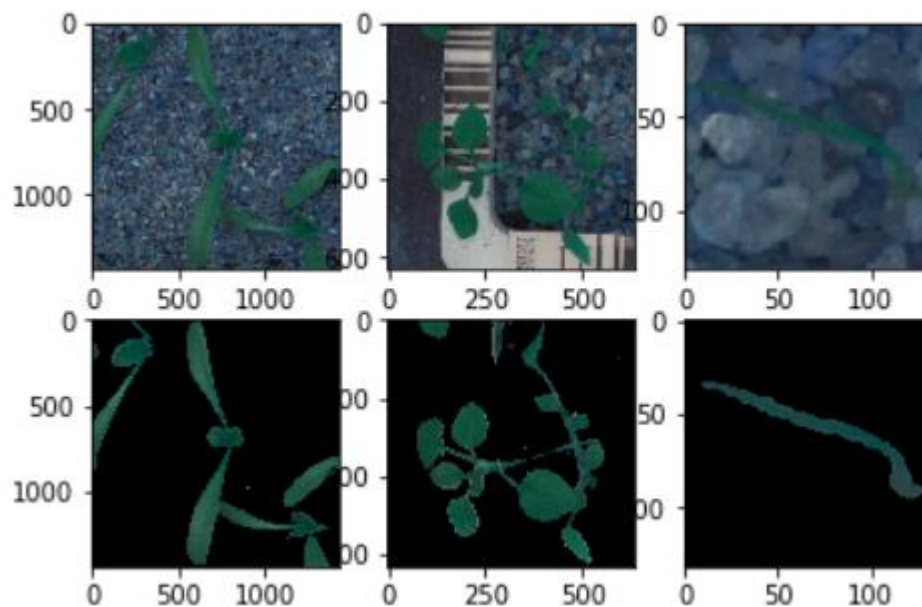
    # Create mask (parameters - green color range)
    lower_green = (25, 40, 50)
    upper_green = (75, 255, 255)
    mask = cv.inRange(hsvImg, lower_green, upper_green)
    kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (11, 11))
    mask = cv.morphologyEx(mask, cv.MORPH_CLOSE, kernel)

    # Create bool mask
    bMask = mask > 0

    # Apply the mask
    clear = np.zeros_like(image, np.uint8) # Create empty image
    clear[bMask] = image[bMask] # Apply boolean mask to the origin image

    # Normalize pixel values
    normalized_image = clear / 255
    return normalized_image
```

The following image shows the original image vs after pre processed image



*** For easier processing we have converted the labels(String format) to a binary classification. So an array of size 12 was created. When a species was detected it was denoted as 1 and vice versa.

Strategy to prevent overfitting

To prevent the overfitting we have created a function that randomly changes the image characteristics during fitting.

```
datagen = ImageDataGenerator(

    rotation_range=180, # randomly rotate images in the range

    zoom_range=0.1, # Randomly zoom image

    width_shift_range=0.1, # randomly shift images horizontally

    height_shift_range=0.1, # randomly shift images vertically

    horizontal_flip=True, # randomly flip images horizontally

    vertical_flip=True # randomly flip images vertically

)

datagen.fit(trainX)
```

Learning Rate Reduction

We intend to reduce the learning rate as the convergence is faster. If the validation accuracy didn't increase for a set no. of epochs (patience=3), then the learning rate is reduced by a factor of 0.4.

```
# learning rate reduction

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',

                                             patience=3,
```

verbose=1,

factor=0.4,

min_lr=0.00001)

Model Evaluation

KFold cross validation was used to evaluate the models. Initially K was set to 5.

```
from sklearn.model_selection import KFold
cv = KFold(n_splits=5, random_state=42, shuffle=False)
k_fold_count = 0
for train_index, test_index in cv.split(X):
    k_fold_count += 1
    print("K FOLD : {}".format(k_fold_count))
    trainX, testX, trainY, testY = X[train_index], X[test_index], TrainLabel[train_index], TrainLabel[
        test_index]
```

QUESTION1

Defining our Model

Our CNN model consisted of 4 convolution layers and 3 connected layers.

- **BatchNormalization** - is used to speed up the learning by normalizing the weights of output of the previous layer.
- **Conv2D** - is used to extract localized image features
- **DropOut** - is used to randomly update dense layer weights to prevent overfitting
- **MaxPooling2D** - is used to reduce the output resolution of the convolution layer.

The summary of the model is as follows.

Model: "sequential"		
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 76, 76, 64)	4864
batch_normalization (Batch Normalization)	(None, 76, 76, 64)	256
conv2d_1 (Conv2D)	(None, 72, 72, 64)	102464
max_pooling2d (MaxPooling2D)	(None, 36, 36, 64)	0
batch_normalization_1 (Batch Normalization)	(None, 36, 36, 64)	256
dropout (Dropout)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 32, 32, 128)	204928
batch_normalization_2 (Batch Normalization)	(None, 32, 32, 128)	512
conv2d_3 (Conv2D)	(None, 28, 28, 128)	409728
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 128)	0
batch_normalization_3 (Batch Normalization)	(None, 14, 14, 128)	512
dropout_1 (Dropout)	(None, 14, 14, 128)	0
conv2d_4 (Conv2D)	(None, 10, 10, 256)	819456
batch_normalization_4 (Batch Normalization)	(None, 10, 10, 256)	1024
conv2d_5 (Conv2D)	(None, 6, 6, 256)	1638656
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 256)	0
batch_normalization_5 (Batch Normalization)	(None, 3, 3, 256)	1024
dropout_2 (Dropout)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 256)	590080
batch_normalization_6 (Batch Normalization)	(None, 256)	1024
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 256)	65792
batch_normalization_7 (Batch Normalization)	(None, 256)	1024
dropout_4 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 12)	3084
=====		
Total params: 3,844,684		
Trainable params: 3,841,868		
Non-trainable params: 2,816		

Confusion Matrix

The confusion matrix is used to analyze the errors in the model and for evaluation purposes. This diagram depicts the confusion matrix we obtained for 12 classes and it is evident that the first class is biased towards 7th class.

```
[[ 0  0  0  0  9  0 91  0  2  0  0  0]
 [ 0 93  1  0  0  3  0  0  5  0  0  0]
 [ 0 16 69  0  7  1  0  2  7  0  0  0]
 [ 0  0  0 94  0  0  1  1  3  3  0  0]
 [ 0  0  1  1 84  0  3  0 13  0  0  0]
 [ 0  1  4  3  1 87  2  0  1  2  0  1]
 [ 0  0  0  0  1  0 99  0  2  0  0  0]
 [ 0  1  0  1  0  0  3 65 32  0  0  0]
 [ 0  2  2  1  1  0 16 14 65  0  0  1]
 [ 0  4  2  9  0  2  0  4 28 49  2  2]
 [ 0 14  0  1  0  0  0  0  3  4 80  0]
 [ 0  0  1  0  0  8  2  1 23  0 10 57]]
```

QUESTION 2

CNN ResNet 50 Model

Two dense layers and one dropout layer was added after the Resnet50 base model. This was changed from the original 1000 classes classification.

```
@staticmethod
def get_model(verbose):
    model = tf.keras.applications.ResNet50(include_top=False, weights='imagenet', input_shape=(80, 80, 3))
    input_layer = model.inputs
    x = model.layers[-1].output
    x = tf.keras.layers.GlobalAveragePooling2D()(x)
    x = tf.keras.layers.Dense(32, activation='softmax')(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(12, activation='softmax')(x)
    model = tf.keras.Model(input_layer, x)
    if verbose == 1:
        print(model.summary())

# compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

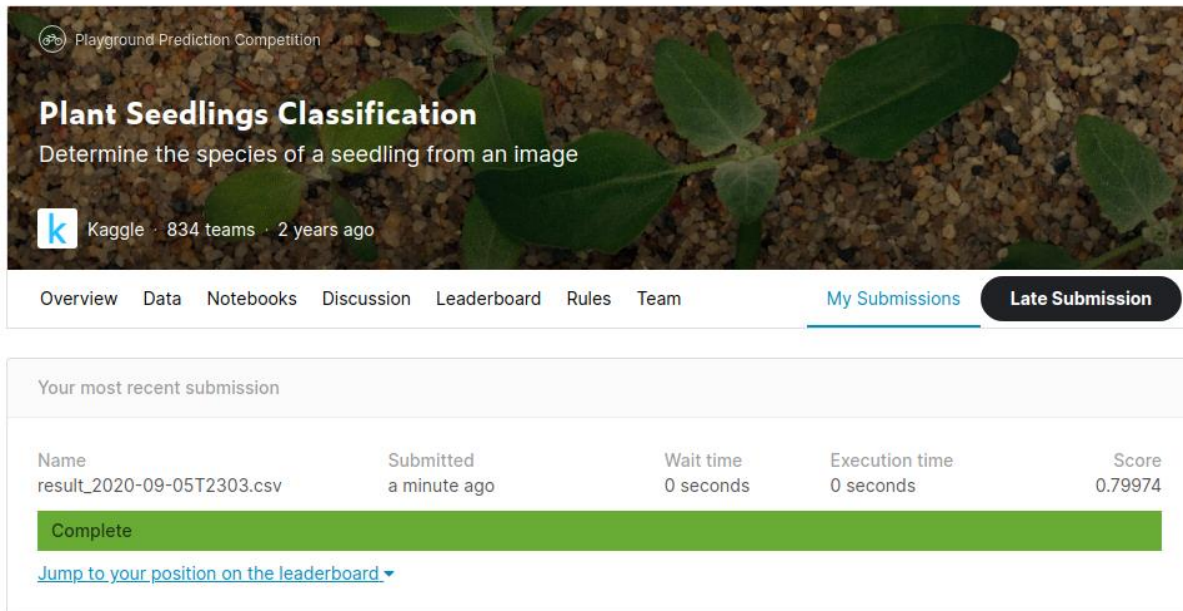
return model
```

This model was replaced with previous model and run the same training again.

QUESTION3

Kaggle submission

We used the predicted data set of our model for the Kaggle submission. The ResNet 50 requires considerable duration of time for execution and requires more training.



Playground Prediction Competition

Plant Seedlings Classification

Determine the species of a seedling from an image

Kaggle · 834 teams · 2 years ago

Overview Data Notebooks Discussion Leaderboard Rules Team [My Submissions](#) [Late Submission](#)

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
result_2020-09-05T2303.csv	a minute ago	0 seconds	0 seconds	0.79974

Complete

[Jump to your position on the leaderboard](#)

REFERENCES

<https://techtutorialsx.com/2019/11/08/python-opencv-converting-image-to-hsv/>

<https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html>

https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau
[u](#)