Project Description – Image classification using CNNs in Keras

Data Description:

You are provided with a dataset of images of plant seedlings at various stages of grown. Each image has a filename that is its unique id. The dataset comprises 12 plant species. The goal of the project is to create a classifier capable of determining a plant's species from a photo.

Dataset:

The dataset can be download from Olympus.

The data file names are:

- images.npy
- Label.csv

The original files are from Kaggle. Due to the large volume of data, the images were converted to images.npy file and the labels are also put into the Labels.csv. So that you can work on the data/project seamlessly without worrying about the high data volume.

The following code was used to convert the large dataset of images to numpy array:

```
# Import necessary libraries.
import math
import numpy as np
import pandas as pd
from glob import glob
data_path = '/content/drive/My Drive/Colab Notebooks/data/plant_seedlings/train.zip'
!mkdir dataset
# Extract the files from dataset to temp_train and temp_test folders (as the dataset is a zip file.)
from zipfile import ZipFile
with ZipFile(data_path, 'r') as zip:
  zip.extractall('./dataset')
path = "/content/dataset/*/*.*"   # The path to all images in training set. (* means include all folder
s and files.)
files = glob(path)

trainImg = []             # Initialize empty list to store the image data as numbers.
trainLabel = []           # Initialize empty list to store the labels of images
j = 1
num = len(files)

# Obtain images and resizing, obtain labels
for img in files:
    '''
    Append the image data to trainImg list.
    Append the labels to trainLabel list.
    '''
    print(str(j) + "/" + str(num), end="\r")
    trainImg.append(cv2.resize(cv2.imread(img), (128, 128)))  # Get image (with resizing to 128x128)
    trainLabel.append(img.split('/')[-
2])  # Get image label (folder name contains the class to which the image belong)
    j += 1

trainImg = np.asarray(trainImg)  # Train images set
trainLabel = pd.DataFrame(trainLabel, columns=["Label"])  # Train labels set
print(trainImg.shape)
print(trainLabel.shape)
trainLabel.to_csv('Labels.csv', index=False)
np.save('plantimages', trainImg)
```

Link to the Kaggle project site: https://www.kaggle.com/c/plant-seedlings-classification/data?select=train

Note: For project purposes, download the data provided on Olympus.

Context:

Can you differentiate a weed from a crop seedling?

The ability to do so effectively can mean better crop yields and better stewardship of the environment.

The Aarhus University Signal Processing group, in collaboration with University of Southern Denmark, has

recently released a dataset containing images of unique plants belonging to 12 species at several growth stages

Objective:

To implement the techniques learnt as a part of the course.

Learning Outcomes:

- Pre-processing of image data.
- Visualization of images.
- Building CNN.
- Evaluate the Model.
- The motive of the project is to make the learners capable to handle images/image classification problems, during this process you should also be capable to handle real image files, not just limited to a numpy array of image pixels.

Guide to solve the project seamlessly:

Here are the points which will help you to solve the problem efficiently:

- Read the problem statement carefully from start to end (including the note at the end). The highlighted part in the attached problem statement should not be missed.
- Download the dataset from the Olympus platform.
- Upload the "images.npy" and “Labels.csv” file to google drive.
- Then you can use the dataset path in the Google Colab notebook to do further steps related to project problem statement.
- You can set runtime type to “GPU” in Google Colab, so that the code will run faster as you will be using CNN to fit your model.

Steps and tasks:

1. Import the libraries, load dataset, print shape of data, visualize the images in dataset. (5 Marks)
2. Data Pre-processing: (15 Marks)
    a. Normalization.
    b. Gaussian Blurring.
    c. Visualize data after pre-processing.
3. Make data compatible: (10 Marks)
    a. Convert labels to one-hot-vectors.
    b. Print the label for y_train[0].
    c. Split the dataset into training, testing, and validation set.
    (Hint: First split images and labels into training and testing set with test_size = 0.3. Then further split test data into test and validation set with test_size = 0.5)
    d. Check the shape of data, Reshape data into shapes compatible with Keras models if it’s not already. If it’s already in the compatible shape, then comment in the notebook that it’s already in compatible shape.
4. Building CNN: (15 Marks)
    a. Define layers.
    b. Set optimizer and loss function. (Use Adam optimizer and categorical crossentropy.)
5. Fit and evaluate model and print confusion matrix. (10 Marks)
6. Visualize predictions for x_test[2], x_test[3], x_test[33], x_test[36], x_test[59]. (5 Marks)

Note:

- **Download the train images from the Olympus Platform.**
- **Do not download the dataset from Kaggle, as:**
    - **The dataset is big.**
    - **The dataset has 2 files for train and test images, but the labels are only for the train file. Test file has no labels associated with it. So, when you want to know the accuracy of model on test images, there’s no way to measure it. That’s why the data provided to you on Olympus has only train images and their labels. For our purpose we use this for our training and testing and validation purpose.**

Happy Learning!