Itroduction to Computer Vision

Plants Seedling Classification- Problem Statement

Due Date:10 Mar 25 3:30 AM

Total Marks:60

Submission Type:File Upload

Description

**Background and Context**

In recent times, the field of agriculture has been in urgent need of modernizing, since the amount of manual work people need to put in to check if plants are growing correctly is still highly extensive. Despite several advances in agricultural technology, people working in the agricultural industry still need to have the ability to sort and recognize different plants and weeds, which takes a lot of time and effort in the long term.

The potential is ripe for this trillion-dollar industry to be greatly impacted by technological innovations that cut down on the requirement for manual labor, and this is where Artificial Intelligence can benefit the workers in this field, *as the time and energy required to identify plant seedlings will be greatly shortened by the use of AI and Deep Learning*. The ability to do so far more efficiently and even more effectively than experienced manual labor could lead to better crop yields, the freeing up of human involvement for higher-order agricultural decision making, and in the long term will result in more sustainable environmental practices in agriculture as well.

**Objective**

The Aarhus University Signal Processing group, in collaboration with the University of Southern Denmark, has provided the data containing images of unique plants belonging to 12 different species. You being a data scientist, need to build a Convolutional Neural Network model which would classify the plant seedlings into their respective 12 categories.

**Data Description**

This dataset contains images of unique plants belonging to 12 different species.

* The data file names are:
  + images.npy
  + Labels.csv
* Due to the large volume of data, the images were converted to numpy arrays and stored in images.npy file and the corresponding labels are also put into Labels.csv so that you can work on the data/project seamlessly without having to worry about the high data volume.
* The goal of the project is to create a classifier capable of determining a plant's species from an image.

**List of Plant species**

* Black-grass
* Charlock
* Cleavers
* Common Chickweed
* Common Wheat
* Fat Hen
* Loose Silky-bent
* Maize
* Scentless Mayweed
* Shepherds Purse
* Small-flowered Cranesbill
* Sugar beet

**Guide to solve the project seamlessly**

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

* Read the problem statement carefully
* Download the dataset from the Olympus platform.
* Upload the "images.npy" and “Labels.csv” files to google drive.
* 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.

**Best Practices for Notebook :**

* The notebook should be well-documented, with inline comments explaining the functionality of code and markdown cells containing comments on the observations and insights.
* The notebook should be run from start to finish in a sequential manner before submission.
* It is preferable to remove all warnings and errors before submission.
* The notebook should be submitted as an HTML file (.html) and NOT as a notebook file (.ipynb)

**Submission Guidelines :**

* 1. There are two ways to work on this project:

**i. Full-code way:**The full code way is to write the solution code from scratch and only submit a final Jupyter notebook with all the insights and observations.

**ii. Low-code way**. The low-code way is to use an existing solution notebook template to build the solution and then submit a business presentation with insights and recommendations.

The primary purpose of providing these two options is to allow learners to opt for the approach that aligns with their individual learning aspirations and outcomes. The below table elaborates on these two options.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Submission type | Who should choose | What is the same across the two | What is different across the two | Final submission file [IMP] | Submission Format |
| Full-code | Learners who aspire to be in hands-on coding roles in the future focussed on building solution codes from scratch | Perform exploratory data analysis to identify insights and recommendations for the problem | Focus on code writing: 10-20% grading on the quality of the final code submitted | Solution notebook from the full-code template submitted in .html format | .html |
| Low-code | Learners who aspire to be in managerial roles in the future-focussed on solution review, interpretation, recommendations, and communication with business |  | Focus on business presentation: 10-20% grading on the quality of the final business presentation submitted | Business presentation in .pdf format with problem definition, insights, and recommendations | .pdf |

Please follow the below steps to complete the assessment. Kindly note that if you submit a presentation, ONLY the presentation will be evaluated. Please make sure that all the sections mentioned in the rubric have been covered in your submission.

**i. Full-code version**

* 1. Download the full-code version of the learner notebook.
  2. Follow the instructions provided in the notebook to complete the project.
  3. Clearly write down insights and recommendations for the business problems in the comments.
  4. Submit only the solution notebook prepared from the learner notebook [format: .html]

**ii.** **Low-code version**

* 1. Download the low-code version of the learner notebook.
  2. Follow the instructions provided in the notebook to complete the project.
  3. Prepare a business presentation with insights and recommendations for the business problem.
  4. Submit only the presentation [format: .pdf]

2. Any assignment found copied/plagiarized with other submissions will not be graded and awarded zero marks.

3. Please ensure timely submission as any submission post-deadline will not be accepted for evaluation.

4. Submission will not be evaluated-

* 1. If it is submitted post-deadline, or,
  2. If more than 1 file is submitted.

**Best Practices for Full-code submissions**

* 1. The final notebook should be well-documented, with inline comments explaining the functionality of code and markdown cells containing comments on the observations and insights.
  2. The notebook should be run from start to finish in a sequential manner before submission.
  3. It is important to remove all warnings and errors before submission.
  4. The notebook should be submitted as an HTML file (.html) and NOT as a notebook file (.ipynb).
  5. Please refer to the FAQ page for common project-related queries.

**Best Practices for Low-code submissions**

* 1. The presentation should be made keeping in mind that the audience will be the Data Science lead of a company.
  2. The key points in the presentation should be the following:
     + Business Overview of the problem and solution approach
     + Key findings and insights which can drive business decisions
     + Business recommendations
     + Focus on explaining the key takeaways in an easy-to-understand manner.
     + The inclusion of the potential benefits of implementing the solution will give you the edge.
  3. Copying and pasting from the notebook is not a good idea, and it is better to avoid showing codes unless they are the focal point of your presentation.
  4. The presentation should be submitted as a PDF file (.pdf) and NOT as a .pptx file.
  5. Please refer to the FAQ page for common project-related queries.

Happy Learning!

Rubric

CriteriaProblem Definition and Data Overview- Define the problem statement - Read the dataset properly - Check the shape of the dataPoints6

CriteriaPerform an Exploratory Data Analysis and get the insights on the images- Plot random images from each of the classes and print their corresponding labels. - Count Plot for each category - Key meaningful observations from EDAPoints8

CriteriaData Pre-processing- Convert the BGR images to RGB images - Resize the images - Plot the images before and after the pre-processing steps - Split the data into train and test - Encode the target variables - Apply the normalizationPoints12

CriteriaModel building- Build Convolution Neural Network - Evaluate the model on different performance metrics and comment on the performance. For example precision, recall, accuracy. - Plot confusion matrixPoints8

CriteriaModel Performance Improvement and Final Model Selection- Build another model using data augmentation to overcome the imbalance problem. - Evaluate the model on different performance metrics and comment on the performance. - Plot confusion matrix - Choose the best model from the ones built with proper reasoning.Points12

CriteriaActionable Insights & RecommendationsConclude with the key takeaways for the businessPoints6

CriteriaPresentation/Notebook - Overall quality- Structure and flow - Crispness - Visual appeal - Conclusion and Business Recommendations OR - Structure and flow - Well commented code - Conclusion and Business RecommendationsPoints8

FAQ

Introduction to Computer Vision

FAQ - Plant Seedlings Classification

[Previous](https://olympus.mygreatlearning.com/courses/117492/modules/items/6639213?pb_id=17873)

[Next](https://olympus.mygreatlearning.com/courses/117492/modules/items/6639215?pb_id=17873)

**Q1. How should one approach the Plant Seedlings Classification project?**

* Before starting the project, please read the problem statement carefully and go through the criteria and descriptions mentioned in the rubric.
* Once you understand the task, download the dataset and import it into a Jupyter notebook or Google Colab to get started with the project.
* To work on the project, you should start with image preprocessing and image visualization.
* Once data is preprocessed, you can use the data to build a model, check its performance based on the desired metric, and if it is not good then train other neural networks.
* You should include all the models that you have trained in your notebook.
* It is important to close the analysis with key findings and conclusions.

**Q2. How to import the dataset on Google Colab?**

You can follow the below steps

1) Download the data from the Olympus ( images.npy and labels.csv ) and upload it into your drive.

2) Mount your google drive using the code below

from google.colab import drive

drive.mount('/content/drive')

3) Reading the dataset using the example code below

import numpy as np  
import pandas as pd  
np.load(path+file\_name )  
pd.read\_csv(path+file\_name )

You should split the data into train and test using the sklearn train\_test\_split function after the necessary preprocessing.

**Q3. How to plot an image from the numpy array?**

You can follow the below steps

1) The images.npy has the images converted into arrays and stored row-wise so each index value represents an image.

2) You can use matplotlib's plot function to plot each image from the numpy array as shown below.

plt.imshow(images[i])

**Q4. What can be done if the given dataset is imbalanced?**

The below code can be used to treat the class imbalance by increasing the weights of the minority classes.

from sklearn.utils import class\_weight  
  
labelList = Labels.Label.unique()  
class\_weights = class\_weight.compute\_class\_weight(class\_weight = "balanced",  
 classes = np.array(labelList),  
 y = y\_train.values.reshape(-1)  
 )  
class\_weights = dict(zip(np.array(range(len(labelList))), class\_weights))  
#print calculated class weights  
class\_weights

**Q5. How to split the images.npy and labels.csv into train and test?**

We can split the data into train and test using the train\_test\_split function as shown below:

from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(images,labels , test\_size=0.1, random\_state=1,stratify=labels)

**Q6. I keep getting the below error while executing the fit function for training CNN. Why?**

ValueError: Shapes (None, 12) and (None, 10) are incompatible

This error is due to the incompatible shape of the output layer. The number of units at the output layer should be equal to the number of labels. Change the Dense output shape into 12 because we have 12 labels in this dataset.

model.add(Dense(12, activation="softmax"))

**Q7. What should the test accuracy of the CNN model be and how should I improve it?**

You can definitely try to decrease the generalization error till the optimal point. There is no specific target accuracy for any real-time problem it depends on the problem as well as the complexity of the dataset.

You can work on the below option to get better performance.

1. Tune Parameters
2. **Image Data Augmentation (Please try this)**
3. Deeper Network Topology
4. Handle Overfitting and Underfitting problem

**Q8. How do you download the HTML file from Google Colab?**

Google Colab does not provide an option to download the HTML version of the code. You can download the .ipynb file from Colab and open this downloaded file in the Jupyter notebook. Jupyter provides the option of downloading an HTML version of your code.

**Q9. Is it mandatory to use Google Colab for this project?**

Yes, it's not mandatory. However, you can try to use either Jupyter Notebook or Google Colab based on your convenience. We prefer using Google Colab for Deep Learning since installing TensorFlow is never an issue.

**Q10. I keep getting the below error while running the code on the Jupyter notebook. Why?**

**ModuleNotFoundError**: No module named 'google.colab'

If you are using a Jupyter notebook, then you need not install google.colab. This library is only used on Google Colab and it comes pre-installed on it.

**Q11. Why do images become black after applying Gaussian blurring and normalization?**

Please apply the Gaussian blurring first and then normalization.

You should use the appropriate function to display the image.

Actually, cv2\_imshow() function plots the image with the 0-1 pixel intensity ( since you have normalized the images ). When you plot an image with 0-1 pixel intensity, it gives black in color. Therefore, either you can multiply 255 while displaying the image or you can use plt.imshow function which does this denormalization internally and plots the image.

**1. Using cv2\_imshow()**

for i in range(5) :  
   cv2\_imshow(X[i]\*255.0)  
   #plt.imshow(X[i])

**2. Using plt.imshow()**

import pylab as plt  
for i in range(5) :  
   plt.imshow(X[i]\*255.0)

**Q12. What should be the input shape to the CNN model if the error is caused due to the input shape and size of the image?**

The input shape to the input layer of the CNN model should be equal to the size of the image.

For example: if the shape of the image is 128x128x3 then the input shape to the first layer of CNN should be 128x128x3

**Q13. What should the order of different layers in a Convolutional Neural Network be? Where should I put my Batch Norm, Dropout, Activation, and Pooling layers? Are there any guidelines regarding the same?**

**Dropout vs Batch Normalization - Standard deviation issue**

There is a big problem that appears when you mix these layers, especially when Batch Normalization is right after Dropout.

Dropouts try to keep the same mean of the outputs without dropouts, but it does change the standard deviation, which will cause a huge difference in the Batch Normalization between training and validation. (During training, the Batch Normalization receives changed standard deviations, and accumulates and stores them. During validation, the dropouts are turned off, the standard deviation is not a changed one anymore, but the original. But Batch Normalization, because it's invalidation, will not use the batch statistics, but the stored statistics, which will be very different from the batch statistics)

So, the first and most important rule is: don't place a Batch Normalization after a Dropout (or a SpatialDropout).

Usually,  try to leave at least two convolutional/dense layers without any dropout before applying a batch normalization, to avoid this.

**Dropout vs Batch Normalization - Changing the zeros to another value**

Also important: the role of the Dropout is to "zero" the influence of some of the weights of the next layer. If you apply a normalization after the dropout, you will not have "zeros" anymore, but a certain value that will be repeated for many units. And this value will vary from batch to batch. So, although there is noise added, you are not killing units as a pure dropout is supposed to do.

**Dropout vs MaxPooling**

The problem of using a regular Dropout before a MaxPooling is that you will zero some pixels, and then the MaxPooling will take the maximum value, sort of ignoring part of your dropout. If your dropout happens to hit a maximum pixel, then the pooling will result in the second maximum, not in zero.

So, Dropout before MaxPooling reduces the effectiveness of the dropout.

**Batch Normalization vs Activation**

Depending on the activation function, using a batch normalization before it can be a good advantage.

For a 'ReLU' activation, the normalization makes the model fail-safe against a bad luck case of "all zeros freeze a ReLU layer". It will also tend to guarantee that half of the units will be zero and the other half linear.

For a 'sigmoid' or a 'tanh', the Batch Normalization will guarantee that the values are within a healthy range, avoiding saturation and vanishing gradients (values that are too far from zero will hit an almost flat region of these functions, causing vanishing gradients).

There are people that say there are other advantages if you do the contrary, I'm not fully aware of these advantages, I like the ones I mentioned very much.

**Dropout vs Activation**

With 'ReLU', there is no difference, [it can be proved that the results are exactly the same (Links to an external site.)Links to an external site.](https://sebastianraschka.com/faq/docs/dropout-activation.html)

With activations that are not centered, such as 'sigmoid' putting a dropout before the activation will not result in "zeros", but in other values. For a sigmoid, the final results of the dropout before it would be 0.5.

If you add a 'tanh' after a dropout, for instance, you will have the zeros, but the scaling that dropout applies to keep the same mean will be distorted by the tanh.

**MaxPooling vs Activation**

There is nothing much here. If the activation is not very weird, the final result would be the same.

**Conclusions:**

Find the appropriate order of layers which is often useful

* Group1
  + Conv
  + Batch Norm
  + Activation
  + MaxPooling
  + Dropout or SpatialDropout
* Group2
  + Conv
  + ----- (there was a dropout in the last group, no Batch Norm here)
  + Activation
  + MaxPooling
  + Dropout or SpatialDropout (decide to use or not)
* After two groups without dropout, can use Batch Norm again