Sign up Sign In

Search Medium Write

Published in Jovian—Data Science and Machine Learning

Soumyajit Behera Follow
Jun 25, 2020 · 5 min read · Listen

Plant — AI

Recognition of Plant Diseases by Leaf Image Classification

61

Lets get started …

Introduction about the project

Food security for billions of people on earth requires minimizing crop damage by timely detection of diseases. Developing methods for detection of plant diseases serves the dual purpose of increasing crop yield and reducing pesticide use without knowing about the proper disease. Along with development of better crop varieties, disease detection is thus a paramount goal for achieving food security. The traditional method of disease detection has been to use manual examination by either farmers or experts, which can be time consuming and costly, proving infeasible for millions of small and medium sized farms around the world.

This project is an approach to the development of plant disease recognition model, based on leaf image classification, by the use of deep convolutional networks. The developed model is able to recognize 38 different types of plant diseases out of of 14 different plants with the ability to distinguish plant leaves from their surroundings.

Step by step approach for building the model

We will be using Pytorch framework for processing of data and building the model.

1.Understanding The Data

The dataset taken is "New Plant Diseases Dataset". It can be downloaded through the link " https://www.kaggle.com/vipoooool/new-plant-diseases-dataset". It is an Image dataset containing images of different healthy and unhealthy crop leaves. The dataset has a Train directory and a Valid directory.

We need to get some basic insights about the directory structure like number of diseases and number of unique plants.

- Finding all the folders present in the train directory

```
os.listdir(data+'New Plant Diseases Dataset(Augmented)/New Plant Diseases Datas
```

```
['Strawberry___healthy',
 'Strawberry___Leaf_scorch',
 'Blueberry___healthy',
 'Potato___Late_blight',
 'Pepper,_bell___healthy',
 'Apple___Black_rot',
 'Tomato___Tomato_mosaic_virus',
 'Peach___Bacterial_spot',
 'Soybean___healthy',
 'Tomato___Early_blight',
 'Grape___Black_rot',
 'Tomato___Septoria_leaf_spot',
 'Squash___Powdery_mildew',
 'Corn_(maize)___healthy',
 'Tomato___Bacterial_spot',
 'Tomato___Target_Spot',
 'Apple___Apple_scab',
 'Tomato___Spider_mites Two-spotted_spider_mite',
 'Cherry_(including_sour)___Powdery_mildew',
 'Corn_(maize)___Northern_Leaf_Blight',
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
 'Corn_(maize)___Common_rust_',
 'Potato___Early_blight',
 'Raspberry___healthy',
 'Grape___healthy',
 'Apple___healthy',
 'Apple___Cedar_apple_rust',
 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
 'Cherry_(including_sour)___healthy',
 'Tomato___Late_blight',
 'Tomato___healthy',
 'Peach___healthy',
 'Orange___Haunglongbing_(Citrus_greening)',
 'Pepper,_bell___Bacterial_spot',
 'Grape___Esca_(Black_Measles)',
```

- Finding the number of unique plants

```
unique_plants = []
cl = os.listdir(data+'New Plant Diseases Dataset(Augmented)/New Plant Diseases
for i in cl:
    x = i.split('_')
    if x[0] not in unique_plants:
        unique_plants.append(x[0])
print("Number of Unique Plants: ",len(unique_plants))
print("Unique Plants: ",unique_plants)
```

```
Number of Unique Plants:  14
Unique Plants:  ['Strawberry', 'Blueberry', 'Potato', 'Pepper,', 'Apple', 'Tomat
'Squash', 'Corn', 'Cherry', 'Raspberry', 'Orange']
```

2.Dataset and Dataloader

Once we are done with checking the directory structure we need to create a dataset and load the images as PyTorch tensors. You need to apply some transforms to the images like resizing them from 256 * 256px to 128 * 128px (for faster processing) and then converting them to Pytorch tensor.

```
transform = transforms.Compose(
    [transforms.Resize(size = 128),
     transforms.ToTensor()])
```

We will load the images from the directory as Pytorch tensors by applying transform to them. We will use the train folder for creating training and validation dataset and valid folder as test dataset.

Once we are done with the dataset creation we will create a training and validation dataset. We will be using 30% of the total dataset as validation data.

We will create a batches of the dataset for faster processing of images. You can use any batch size of your choice as long as it fits into memory.

3.Building the CNN Model Architecture

Now comes the main part i.e Building the Model. Initially we define a base image classification class which will have functions for training, and validation of each batch of data . This will help us to build several CNN models without writing this functions again and again.

We will inherit the base class and build CNN Architecture.

I have given example of 2 CNN architecture:

> *Building a multi layer CNN model from scratch using Conv2D, Relu and Maxpool2D and Linear Layers.*

> *Building a CNN model using Transfer Learning. Below is an example of transfer learning using Resnet34.*

4.Loading the data into GPU Device and Creating the Model

We need to load the data form CPU to GPU for faster processing of the images. As the data is very large computing on CPU will take a huge lot of time. So we use GPU.

The below methods help in selecting the device and moving the data from CPU to GPU(if available).

Now we will load the training, validation and test dataloader into GPU using `DeviceDataLoader` function.

Then we create a model using the instantiating the `Plant_Disease_Model2` class.

5.Training and Evaluation

Now its time to train our model and fit the data into it. We will define two functions `evaluate` and `fit` for evaluation and training of the model.

Then we call the `fit` function with its different parameters to train the model for certain number of epochs. We use optimizer *“Adam”* and and Learning Rate of *0.001* to train the model.

6.Evaluating on test data and Making prediction

Once we are done with the training of the model we need to evaluate our model to know how goods our model performs with unseen test data. We will call the `evaluate(model,test_loader)` method with model and test data.

We define a `predict_image` function to generate prediction for a single image.

We call the `predict_image` function on a single image of test data.

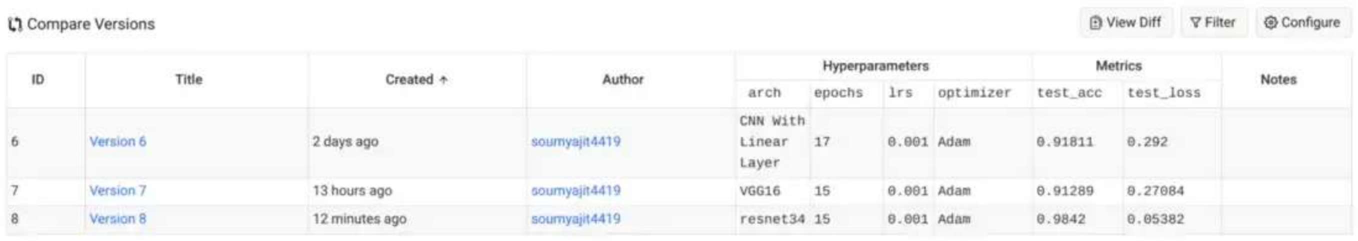
The model gives a correct prediction for the image.

7.Saving the model

Final step, we save the weights to the model so that we can use the pre-trained weights , instead of retraining it again and again.

Trying Various Models

You can try building various CNN models and using various hyperparamers like changing the Optimizer, Lr and Applying some image argumentation techniques to increase the accuracy of the model. Blow are some models tried by me:


Compare Versions
View Diff
Filter
Configure

ID	Title	Created ↑	Author	Hyperparameters				Metrics		Notes
				arch	epochs	lrs	optimizer	test_acc	test_loss	
6	Version 6	2 days ago	soumyajit4419	CNN With Linear Layer	17	0.001	Adam	0.91811	0.292	
7	Version 7	13 hours ago	soumyajit4419	VGG16	15	0.001	Adam	0.91289	0.27084	
8	Version 8	12 minutes ago	soumyajit4419	resnet34	15	0.001	Adam	0.9842	0.05382	

Various CNN architecture

Conclusion

- We were successfully able to train a image classification model using various CNN architecture with an test accuracy of 98.42%.

> *Further Work*

- We can the use image localisation techniques to find the exact location of the affected area of the leaf.
- We can build a Flask API for deploying this model for the use in real life.

Thanks For Reading……

Thanks for reading

References:

1. Official documentation:- PyTorch Documentation
2. Jovian Notebook:- My notebook
3. Github Repo:- Repository

> **Any help needed , find me on twitter or linkedIn , Or Visit my website at mywebpage , for any query feel free to drop a mail at ashish454570@gmail.com!!**