Capstone Project

Rairo Mukamuri

June 8th, 2020

Machine Learning Engineer Nanodegree

Definition

Project Overview

As the world grows in population every year the demand for food is also rising, however a growing threat in the form of plant diseases threatens to worsen the already strained food supply.

Pest and disease induced crop losses have devastated farms across the world but thanks to advances in machine learning farmers can detect diseases early through the use of computer vision in the browser or on a mobile device. Convolutional Neural Networks which are a class of deep neural networks for analysis visual imagery have been widely used in image classification problems and their applications have been promising.

Problem Statement

The goal of this project is to fit a Convolutional Neural Network on the data using the Fastai library and Pytorch. I am going to use transfer learning with ResNet, a deep residual learning framework developed at Microsoft in order to achieve accurate classification of the plant images. Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Resnets have been trained on the imagenet dataset which contains millions of images so their knowledge of those features will be very helpful in this case.

Metrics

The model will be evaluated on accuracy, a common metric for classification problems; it takes into account both true positives and true negatives with equal weight.

$$\textit{accuracy} = \frac{\textit{true positives} + \textit{true negatives}}{\textit{dataset size}}$$

Analysis

Data Exploration

The PlantVillage dataset consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease.

>>plant_data.classes

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

Some of the images from the dataset

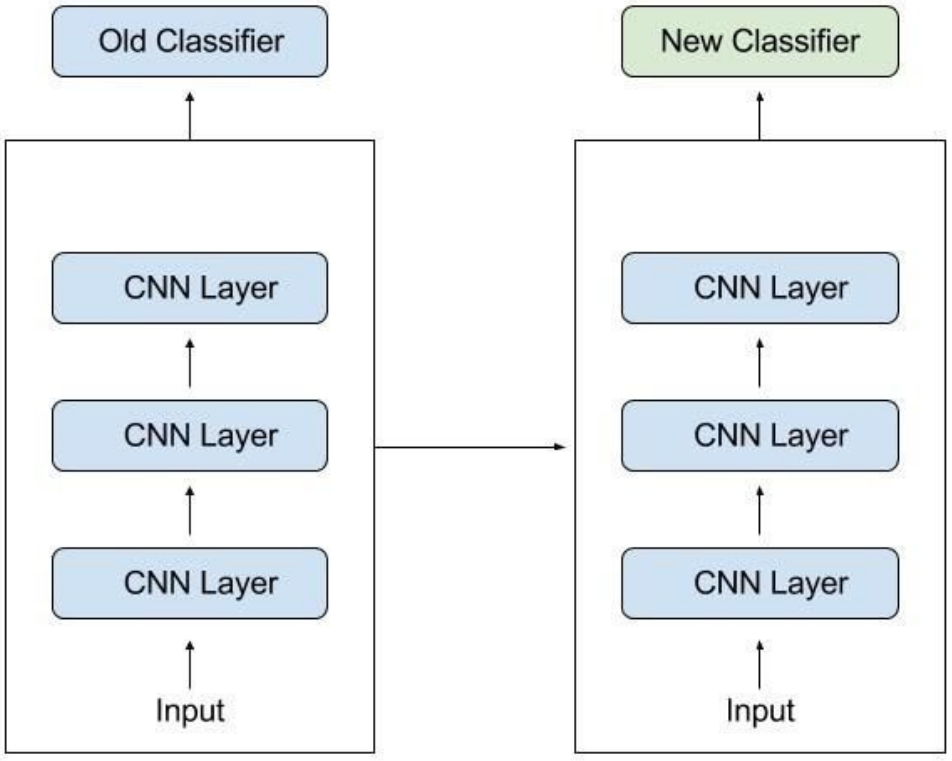
Fig. 1 Images from the plant village dataset

Algorithms and Techniques

The classifier is a Convolutional Neural Network,which is the state-of-the-art algorithm for most image processing tasks, including classification. The best way to tackle this problem, I presumed, would be to use transfer learning with resnet50 and resnet34 and investigate their performance on the data.

Some of the main benefits of transfer learning are reduction of computational time, processing power and data requirements. Knowledge of previously learned features will be passed on to our classifier.

Fig. 2 Transfer Learning



Benchmark

For this problem, the benchmark model will be a resnet-152 pytorch classifier developed by Viridiana Romero Martinez as referenced in her medium article https://medium.com/datadriveninvestor/creating-an-ai-app-that-detect-diseases-in-plants-using-facebooks-deep-learning-platform-pytorch-15faaeb6bec3 . The model produced a high accuracy score, 96.1% and the goal is to see if fastai and another resnet can improve on that.

Methodology

Data Preprocessing

Preprocess the data to make it easier for train by resizing every image to (224, 224) and then use the `.normalize(imagenet_stats)` method from fastai to normalise the dataset based on the stats of the RGB channels from the ImageNet dataset

```
>>plant_data = ImageDataBunch.from_folder(path=PATH, train='Train',
valid='Test', ds_tfms=get_transforms(), size=224, bs=bs)
plant_data.normalize(imagenet_stats)
```

```
ImageDataBunch;

Train: LabelList (43429 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3,
224, 224),Image (3, 224, 224)
y: CategoryList
Apple___Apple_scab,Apple___Apple_scab,Apple___Apple_scab,Apple___Apple_
scab,Apple___Apple_scab
Path: PlantDiseaseData;

Valid: LabelList (10876 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3,
224, 224),Image (3, 224, 224)
y: CategoryList
Apple___Apple_scab,Apple___Apple_scab,Apple___Apple_scab,Apple___Apple_
scab,Apple___Apple_scab
Path: PlantDiseaseData
```

Implementation

The implementation process can be split into two main stages:

1. The classifier training stage

2. The application development stage

During the first stage, the classifier was trained on the preprocessed training data. This was done in two Jupyter notebook using Google colab (titled “resnet_50” and “resnet_34”), and can be further divided into the following steps:

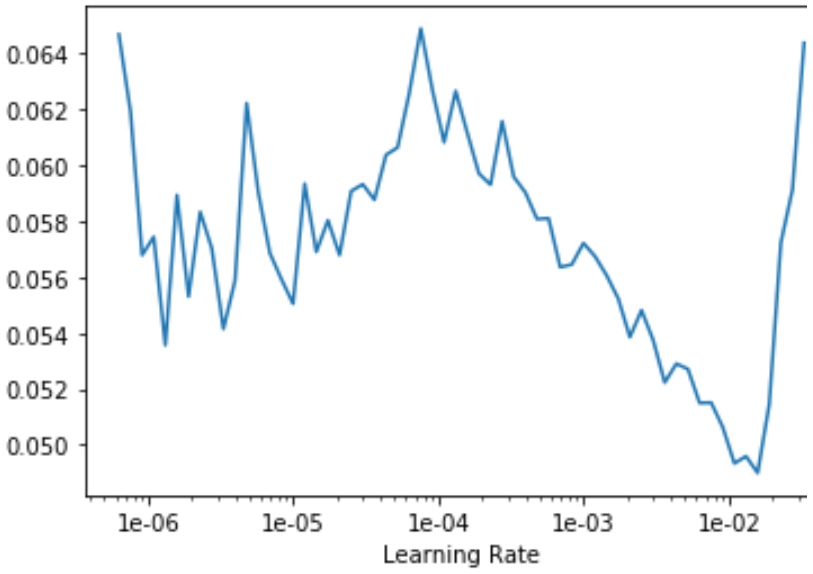
1. Load both the training and validation images into memory, preprocessing them as described in the previous section
2. Choose a pretrained model
3. Define the loss function, accuracy
4. Train the network, logging the validation/training loss and the validation accuracy
5. Plot the logged values
6. Find The learning rate
7. Train again
8. Save and freeze the trained network
9. Export the model as a serialized object

The Application development stage involves deploying the model in a simple python web app hosted https://render.com/ where a user can upload images of plants and the web app will classify them.

Refinement

The Hyperparameter tuning involves finding the learning rate after a few cycles, typically the learning rate is the steepest point on the gradient descent curve but the `slice` method allows us to use a range to be more accurate.

Fig. 3




```
model.unfreeze()
model.fit_one_cycle(3, max_lr=slice(1e-03, 1e-02))
```

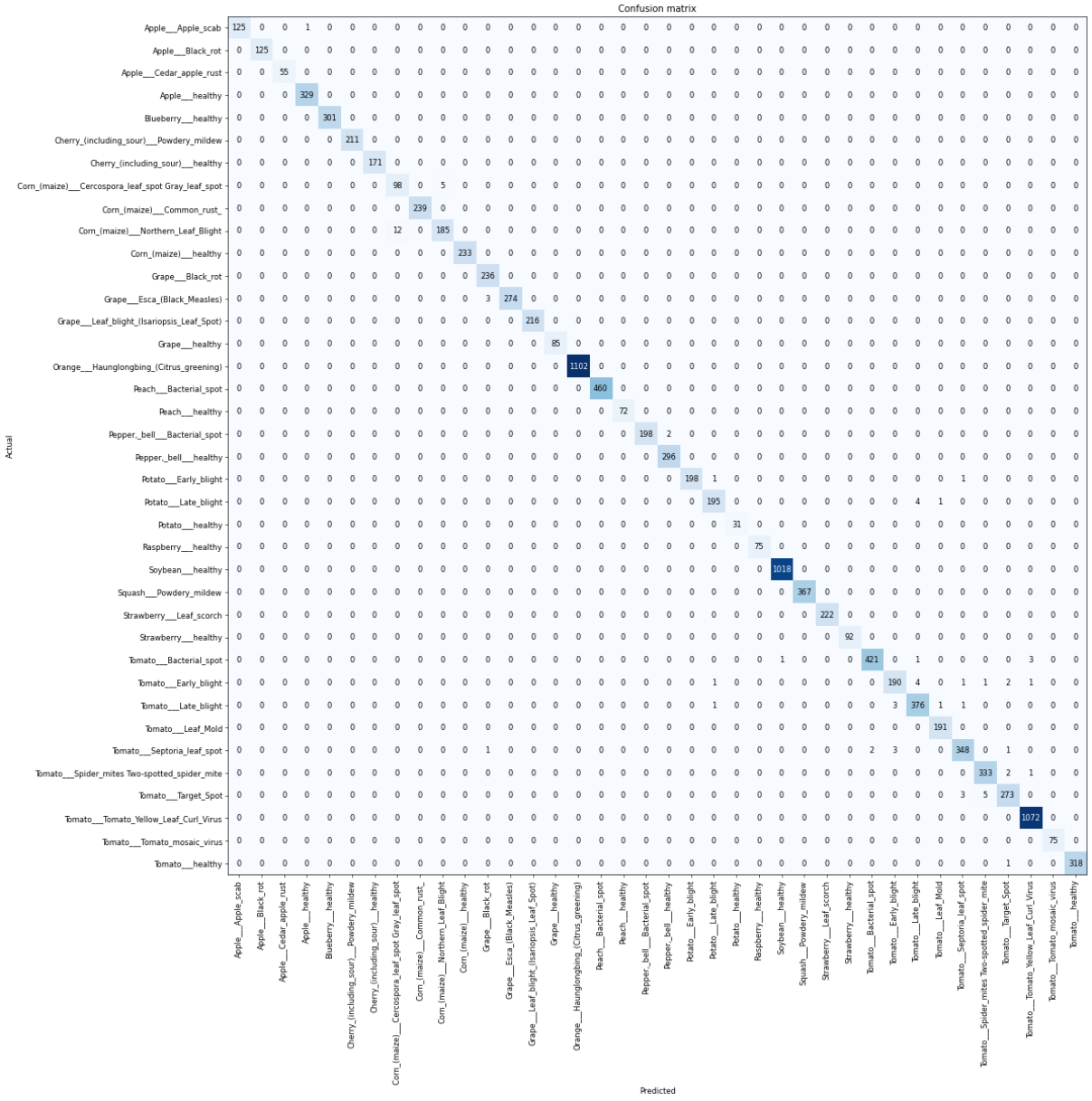
Results

Model Evaluation and Validation

Both models resnet50 and resnet34 performed very well in the transfer learning exercise with very few losses.

Fig. 4
Some losses

Fig. 5
Confusion matrix



Justification

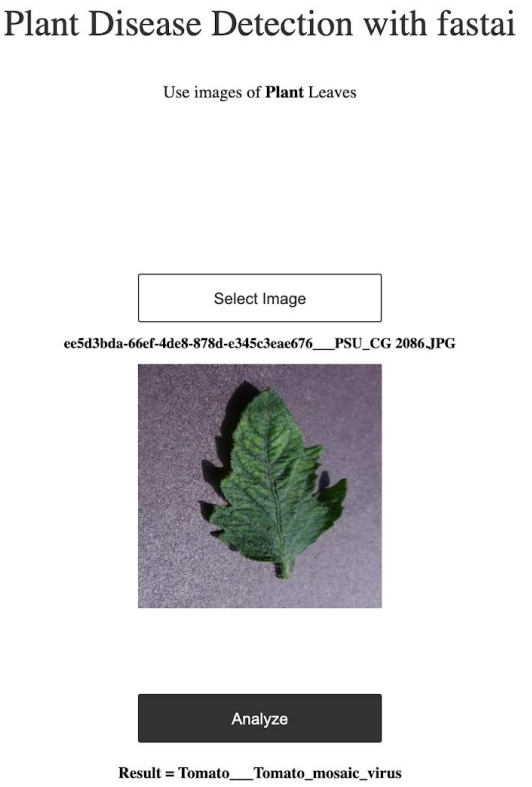
Resnet34 had an accuracy of 99.67% , Resnet50 had an accuracy of %99.30 both higher than the benchmark which was 96.1%

The application runs well, is simple to use and is quite accurate among full leaf images.

Conclusion

Free-Form Visualization

Fig. 6 Example of the web app in action



Reflection

The process used for this project can be summarized using the following steps:

1. **Data exploration and preprocessing:** Explore the data by looking at the number of classes and then preprocess it to make it easier for train by resizing every image to (224, 224) and then use the .normalize(imagenet_stats) method from fastai to normalise the dataset based on the stats of the RGB channels from the ImageNet dataset.

2. **Model selection:** Choose a pre-trained model starting with Resnet34 and Resnet50.

3. **Training and validation:** Train the convolutional neural network on the training set and then fit it on the test/validation set.

4. **Summarisation of the results:** Calculate the accuracy and construct a confusion matrix Develop and deploy a web application: Use python and flask to develop a simple web application for a user to input images which will be classified by the model and return the results to the user.

I found steps 4 to be challenging the most as I had to learn how to quickly deploy a fastai model, I eventually found render.com to be the quickest and easiest way to do it.

Improvement

Addition of more variable data will improve the model as it is only comfortable with close up leaf images. Deploying it to mobile devices using tensorflow lite will also make the application more user friendly as the user will readily use the phone camera.

References

https://medium.com/datadriveninvestor/creating-an-ai-app-that-detect-diseases-in-plants-using-facebooks-deep-learning-platform-pytorch-15faaeb6bec3

https://builtin.com/data-science/transfer-learning

https://github.com/MichaelGerhard/PlantDiseaseData

David Hughes paper, https://arxiv.org/ftp/arxiv/papers/1604/1604.03169.pdf

https://neurohive.io/en/popular-networks/resnet/
https://towardsdatascience.com/create-and-deploy-an-image-classifier-using-fastai-and-render-in-15-mins-947f9de42d21