Plants Diseases Classification with ResNest Architecture

Mandelli Lorenzo 1747091

1 Introduction

Project Overview

Plant diseases are considered one of the main factors influencing food production and minimize losses in production. The recent expansion of deep learning methods has found its application also in plant disease detection, offering a robust tool with highly accurate results. The project aims to comprehend and implement a ResNest-like CNN model for plant diseases classification tasks. I decided to work on this task because I have already worked about it, trying to find a solution based on a sequential CNN, and than because, studying the proposed paper, I found out that the ResNest architecture could in fact be a very interesting alternative/better solution for the task considered. I planned to implement both a sequential (SEC) and a RES (RES) model based on the multi-path attention mechanism proposed by paper authors and compare the results considering as a baseline (BASE) my old sequential CNN.

Split-Attention Network

The neuron connections in visual cortex have inspired the development of CNNs in the past decades. The method proposed by the authors captures cross-channel feature correlations. This is done by the concept of grouped convolution. The authors propose the so called Split-Attention Block which performs a set of transformations on low dimensional embeddings and concatenates their outputs as in a multi-path network. Each transformation incorporates channel-wise attention strategy to capture interdependencies of the featuremap. This mechanism models the interdependencies of featuremap channels, which uses the global context information to selectively highlight or de-emphasize the features. This attention mechanism is similar to attentional selection stage of human primary visual cortex, which finds the informative parts for recognizing objects. Human/animals perceive various visual patterns using the cortex in separate regions that respond to different and particular visual features. This strategy makes it easy to identify subtle but dominant differences of similar objects in the neural perception system. [1]

Experiment

In order to successfully complete the project I designed a small experiment. The main purpose of this experiment is to prove that RES models works better than SEC and BASE models, both in terms of computational costs and performances achieved, and that augmentation techniques enforce the generalization of the achieved prediction power. Therefore I repeated the training for each model both for RES and SEC cases, with and without any preprocess technique previously applied to the data. I collected the results, plots accuracies, losses and put them together in order to compare them. After this initial process, I used a little hand crafted dataset which contains images taken by me and some friends, in order to evaluate the best models over REAL data. Since it is a multiclass classification task I do not expected to obtain high accuracies; it is also true that I had only some of the classes the original dataset posses.

2 Dataset

I used a kuggle dataset containing around 300,000 images in a 256 x 256 pixels format, belonging to 38 classes describing 14 plants and 19 diseases. Each image contains one leaf with a contrasting background. Leaves are not always vertically aligned and not always in focus. Since the dataset is very large I decided to use just a portion of it ; in order to do so I selected for each class a subset of images, and than according to 60-20-20 criterion I have splitted the resulting subset into train, validation and test set. I ended up with a subset of the original dateset consisting in 54,305 samples, where 32,571 have been used as train set, 10,858 as validation set and the remaining 10,876 as test set.

3 Implementation: Sequential and RES Models

Both for SEC and RES models I started working around with a ResNest-50 like architecture thought to be trained with the well know ImageNet dataset. This kind of architecture is composed by 4 layers built by stacking ResNest-Convolution blocks and ResNest-Identity blocks. RES and SEC models differs just for shortcut connection between layers. This shortcuts or highways are realized by simply adding information coming from previous layers to next layers via identity blocks made by unitary convolutions; in this way the information can flow across multiple layers without being modified.

Hyper Parameter Tuning and Models Selection

Since the considered CNN is thought to be trained on a very large dataset and used to predict over a huge number of classes, the architecture brings the numbers of features to 2048 with a very small dimension (4x4). After some first training, I noticed, in fact, the network was considerably oversized, with more than 19 million parameters, which explode towards the last fully connected layers. It is indeed clear that bringing the numbers of features to 2048 with such a small dimension (4x4) is unnecessary if not incorrect. Than I reshaped the network dimensions, both width and depth, in order to best fit my task. Therefore I tryed using only 1, 2 and 3 layers resulting in three different architectures. The model with just the first layer, immediately resulted to be undersized, than I did not considered it for further use. I also played a bit with some of the layer hyper-parameters such as the number of kernels or the number of feature-channels; this last one is controllable by the group number in conv2D layers. I do not tryed changing kernel dimensions since it would have ended up in a far too invasive modification. The best resulting architectures are model-2 and model-3. About this last kind of architecture, I also changed and tryed using every relevant possible layer combinations like: 2-2-5 , 2-3-2 or 3-4-3 and for each of them I also modified the learning head of the net. This hyper-parameter tuning procedure ended up showing that the best combination is given by model-2 or model-2-3-2 or model-3-4-3 with two-groups convolutions and a 256 units dense layer followed by a dropout layer and the 38 unit softmax layer. Obviously each of this architectures has different sizes in terms of depth and number of trainable parameter and resulted in different performances but always over 0.9 accuracy and under 0.15 loss.

Additional Implementation Details

For the model implementation, model training and model testing, I used the Keras package from Tensorflow library. In order to correctly implement the proposed architecture I composed the network by using three different blocks, respectively, res-conv1, res-conv2 and res-identity; each of these blocks implement a split-attention-like block, using a grouped convolution. The res-conv1 is used just for building the first layer; it applies a smaller down sampling operation (AveragePooling with unitary kernel and unitary stride). The res-conv2 is used for building all the remaining layers and apply a wider down sampling operation (AveragePooling layer with kernel and stride dimension of (2,2)). The res-identity instead do not implement any down sample operation. Each layer is built stacking a res-conv1 or a res-conv2 block followed by one or multiple res-identity blocks and adding a shortcut connection between each block. The sequential model is implemented following the same exact criterion, with the only difference that shortcut connections are removed and layer are simply stacked one after the other.

4 Baseline and Base Preprocess

As previously mentioned I will use a CNN I already posses as a baseline. This model is a VGG-like architecture resized for the chosen task. As a good practice for this kind of ML tasks, I Considered using some data-augmentation techniques to let the model possibly learn more general features while training; Since Keras do not provide any cropping function, than I implemented one. Together with the crop function I decided to try the following Keras preprocessing functions: vertical and horizontal flip (VF and HF), rotation range (RR) and brightness range (BR). From my previous work on the BASE model I knew that the combination of vertical and horizontal flips, rotation range up to 10 degree and brightness range between 0.2 and 1.0 is actually one of the best combinations; in fact it resulted in The models trained with data augmented in this way has been labeled as model-AUG. In Table1 are displayed the results collected evaluating the BASE models trained with plain data and with different kind of augmented data.

BASELINE: PLAIN vs AUG models		
Structure	Accuracy	Loss
model-Plain	0.9293	0.2296
model-AUG0	0.9335	0.2143
model-AUG1	0.9352	0.2129
model-AUG	0.9379	0.2115

Table 1: The 3 data augmentation above AUG0, AUG1, AUG are respectively VF + HF, VF + HF + BR and VF + HF + BR + RR where, VF and HF stands for vertical and horizontal flips, BR stands for brightness range and RR stands for rotation range. As we can see data augmentation techniques also improve a bit performances achieved over the test set obtained as a partition of the dataset.

5 Results : Tables and Plots

The results I collected show that the RES models reaches high accuracy seriously faster than SEC models; typically RES models hits 0.9+ accuracy within 5 to 10 epochs while SEC ones toward or after the 20th epoch as figure2 and figure3 show. Comparing figure 3 with 4 It is observable how models trained with plain data converges faster than models trained with augmented one. We can also notice how REC models always reaches higher accuracies and lower losses than SEC models, confronting figure 1 or 2 with figure 3 or 4.

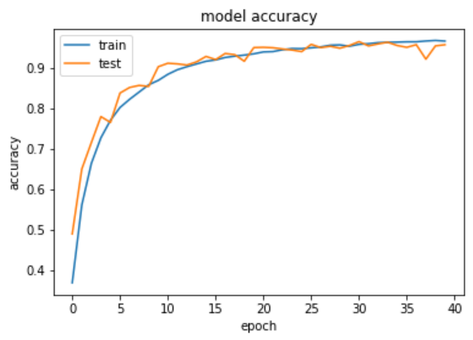

Figure 1: Accuracy plots for SEC model-2L-PLAIN

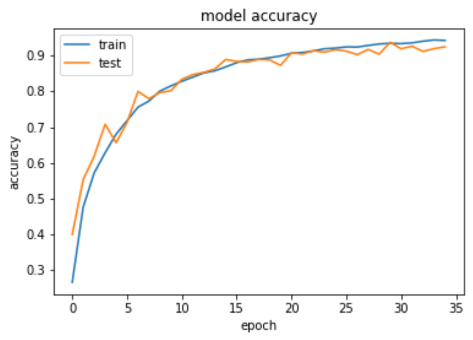

Figure 2: Accuracy plots for SEC model-3-4-3-PLAIN

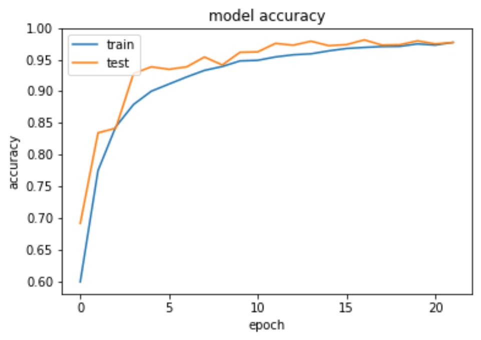

Figure 3: Accuracy plots for RES model-2-3-2-PLAIN

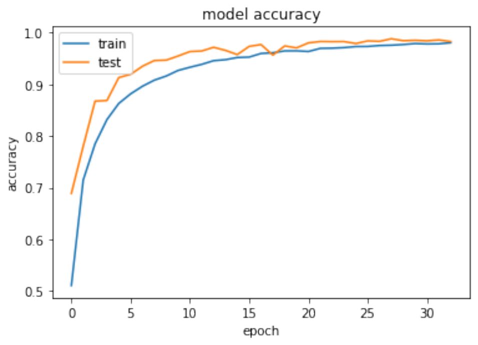

Figure 4: Accuracy plots for RES model-2-3-2-AUG

Generally I could say that data augmentation did not discriminate between performance differences for SEC while it did for RES models as we can observe looking at table 2 and 3. The most evident case is the model-2-3-2 which; Accuracy REAL passes from 0.2963 to 0.4444 when AUG is applied, table 3.

SEC models models PLAIN (P) vs AUG (A)		
Structure	Accuracy TEST	Accuracy REAL
model-2L-P	0.9750	0.3541
model-2L-A	0.9750	0.3541
model-2-3-2-P	0.9553	0.2407
model-2-3-2-A	0.9555	0.2999
model-3-4-3-P	0.9453	0.3519
model-3-4-3-A	0.9471	0.3519
model-2-2-5-P	0.8905	0.2593
model-2-2-5-A	0.8917	0.2593

Table 2: SEC models trained with AUG data shows very little improvements apart from model-2-3-2.

RES models PLAIN (P) vs AUG (A)		
Structure	Accuracy TEST	Accuracy REAL
model-2L-P	0.9815	0.2963
model-2L-A	0.9889	0.2963
model-2-3-2-P	0.9821	0.2963
model-2-3-2-A	0.9843	0.4444
model-3-4-3-P	0.9844	0.2963
model-3-4-3-A	0.9913	0.3148

Table 3: RES models trained with AUG data shows significant improvements of performances both when evaluation is done over TEST and REAL set.

A final comparison has been done collecting all the relevant information into the table 5, which highlights how RES models performs significantly better than SEC ones and also how data augmentation techniques can eventually let the model learn more general features.

Final Comparison: BASE vs SEC vs RES P vs A		
Structure	Accuracy TEST	Accuracy REAL
Base-P	0.9293	0.2203
Base-A	0.9356	0.2485
SEC-2L-P	0.9750	0.3541
SEC-2L-A	0.9750	0.3541
RES-2-3-2-P	0.9821	0.2963
RES-2-3-2-A	0.9843	0.4444