Large-Scale Plant Classification using Deep Convolutional Neural Networks

Josef Haupt[1], Stefan Kahl[1], Danny Kowerko[2], and Maximilian Eibl[1]

[1] Chair Media Informatics,
Chemnitz University of Technology, D-09107 Chemnitz, Germany
[2] Junior Professorship Media Computing,
Chemnitz University of Technology, D-09107 Chemnitz, Germany
{josef.haupt, stefan.kahl, danny.kowerko,
maximilian.eibl}@informatik.tu-chemnitz.de

Abstract. Deep learning techniques have significantly improved plant species classification in recent years. The goal of the 2018 ExpertLife-CLEF challenge was to compare the performance of human experts to machines trained on the PlantCLEF 2017 dataset containing 10.000 classes. We used the Inception, ResNet and DenseNet architectures to solve this complex task. In our experiments, complex neural net layouts yield strong results, comparable to human performance. We further push the overall accuracy through iterative adjustment of class weights. An ensemble consisting of a ResNet50 and two DenseNet201 with fine-tuned class weights reached a top1-accuracy of 77% on the test set.

Keywords: Deep Learning · Plant Classification · Convolutional Neural Networks

1 Introduction

The ExpertLifeCLEF 2018 challenge [1] is the continuation of last year's PlantCLEF 2017 task and part of LifeCLEF 2018 [2]. The main goal was to compare the performance of human experts and machines. In this paper we are going to describe our approach, the model architectures we used and our training process.

2 Dataset

The dataset provided by CLEF is split into two parts. One part consists only of trusted images from the Encyclopedia Of Life (EoL), which implies that the object shown is a plant and is also labelled correctly. The second dataset has been built using Bing and Google search engines with no further validation in place [3]. This means, the number of samples per class is much higher, but images might not be labeled correctly or do not contain any plants at all.

This year's training set is exactly the same as for the PlantCLEF 2017 task. The clean dataset holds 256.288 samples and the noisy set 1.432.162. These samples are for 10.000 different plants. Both datasets show massive class imbalance.

2.1 Dataset Preparation

No changes have been made regarding the clean set. Some images from the noisy set had to be removed since they were either not compatible with image processing tools or/and had corrupt EXIF data, which led to complications. This affected 4.398 total images which is a very small part of the whole dataset and therefore neglectable. No further filtering has been done to either of the datasets.

3 Experiments

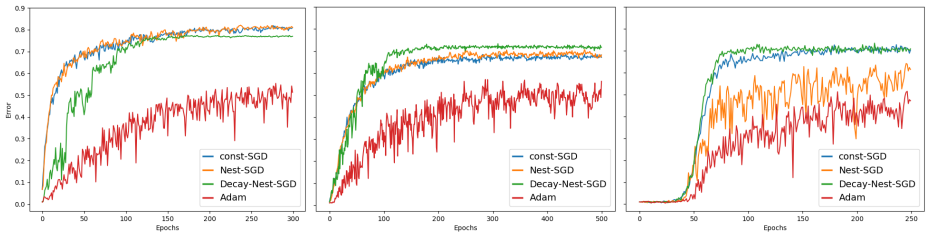
We tested and used three network architectures, namely Inception v3[4], ResNet50[5] and DenseNet201[6]. We performed most of our experiments using the DenseNet model which seemed to perform much better than the Inception v3 architecture and slightly better than the ResNet50 model. Additionally, DenseNet is a relatively recent architecture which was not used in the previous years (see e.g. [7]).

3.1 Fine-tuning

The mentioned three CNNs have been tested with various learning rates, batch sizes and optimizers to find the best hyperparameter setup. All experiments described have been done using the clean dataset.

Batch Size We used a NVIDIA P6000 graphics card for training. Since it was not possible to utilize all of the 24GB of VRAM for the training of a single model, the GPU memory was split into parts of 8GB. This way, we were able to train multiple models at the same time and finish more experiments. The restriction of VRAM per model led to a maximum batch size of 64 for the Inception v3 which outperformed models trained with a lower batch size. The ResNet50 had similar results, but with a maximum batch size of 32. The DenseNet201 used even more of the GPU memory and could only be used with a batch size of 16.

Optimizer Inception, ResNet and DenseNet were each tested with Stochastic Gradient Descent (SGD) and Adam. A static learning rate of 0.01 was used for the SGD. All architectures performed much better with the SGD than with then the adaptive optimizer.



(a) DenseNet201 (b) ResNet50 (c) InceptionV3

Fig. 1: This figure shows the results of some of our experiments regarding the optimizer.

Learning Rate Since the SGD optimization was much more successful than Adam, we tried to increase the performance with further fine-tuning. We tested all networks with three different settings for the SGD:

- a static learning rate of 0.01
- a static learning rate of 0.001 and 0.9 Nesterov momentum
- a decaying learning rate schedule with 0.9 Nesterov momentum

The decaying learning rate schedule calculates the learning rate after every epoch using the following equation:

$$lr = initial * drop\_rate^{\lfloor \frac{1+epoch}{drop\_epoch} \rfloor} \tag{1}$$

The objective was to divide the learning rate by half after every tenth of the planned maximum epochs. Which means $drop\_epoch$ calculates from $max\_epochs/10$.

Table 1: The final settings for the used architectures.

architecture	batch size	optimizer	learning rate
Inception V3	64	SGD	decay
ResNet50	32	SGD	decay
DenseNet201	16	SGD	0.001-0.0001

The experiments showed that the decay schedule was best for the Inception v3 and the ResNet50. The DenseNet201 did not benefit from this schedule and performed better with a static learning rate and Nesterov momentum.

3.2 Data Augmentation

We used a variety of augmentation operations to further increase the dataset diversity. Selected experiments showed that the validation error can be greatly

(a) Original. (b) Zoomed in. (c) Rotated. (d) Height shift.

Fig. 2: Examples of the used augmentations.

reduced by the augmentations. Finally, we decided to implement following augmentation methods: Horizontal and vertical flip, zooming, rotating, shearing and shifting.

The range of every augmentation method was determined by empirical testing. Augmentation artifacts, for example as result of rotation, were filled in with a reflection of the original image content.

4 Training

We trained the ResNet and Inception models according to best practices. However, we tried to fine-tune the DenseNet201 with adjusted class weights over multiple iterations and we will describe this specific process in more detail. For all experiments, ten percent of the trusted image set were used for validation.

4.1 Fine-Tuning Class Weights

In an attempt to balance the classes, the initial weights were generated from the initial distribution of the samples. Therefore, the weight of each class was initially calculated using the average samples per class divided by the number of samples of that particular class. This redistribution of class weights already improved the accuracy by 9% over our local test set. Over the course of six iterations, the class weights were further modified. With each iteration a small test set was predicted and the error rates per class were calculated. The error rate is the percentage of samples which were classified correctly.

We modified each class weight using the following equation:

$$weight = weight * (1.85 - error\_rate) \tag{2}$$

We reduced weights for classes with an accuracy of 85% or higher by a small percentage; classes with a lower accuracy got higher weights in the next iteration.

4.2 Training Process

Each model was trained with 10.000 samples per epoch, for up to 1.000 epochs. Even with a very powerful GPU, the training of one model took three to five

days. After the training, new class weights were generated using the error rates of the model on the test set and another model was trained with the modified class weights. We repeated this process six times until we submitted our runs.

5 Results

This year's ExpertLifeCLEF challenge was themed as human experts vs machines. The results show that some of the experts did have a remarkably high accuracy, which was rather surprising - but not all experts did beat the machine performance. Our system was able to beat one expert and scores were within a small margin with other experts.

Most of our models were trained on the noisy data set, because the last year's PlantCLEF results showed that the highest scores were achieved when training mostly on the noisy data [3]. We submitted five runs, but run 1 and 2 are only marginally different. Therefore, run 2 will not be discussed here.

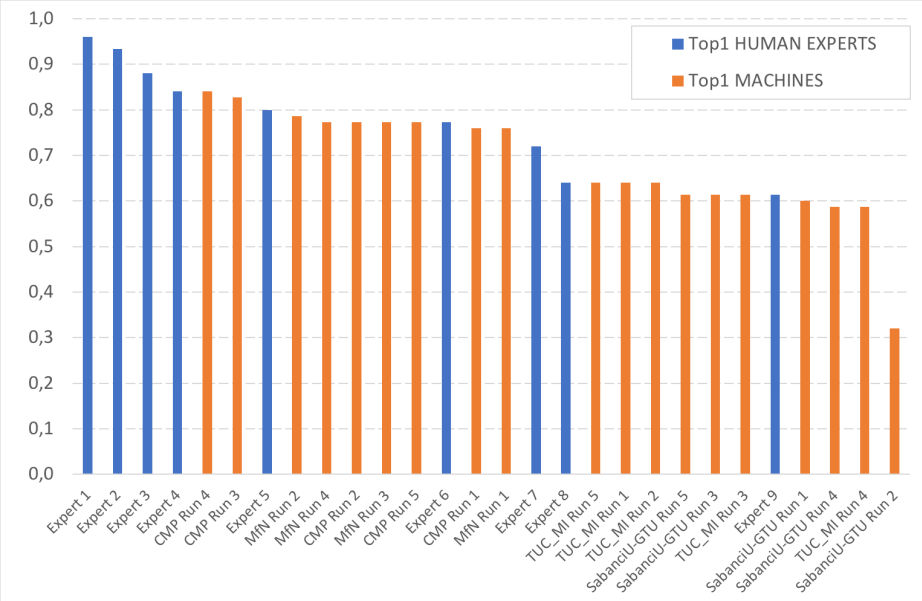


Fig. 3: Official ExpertLifeCLEF scores. Machine learning systems outperform the majority of the human experts. However, some of the human expert are still leading with scores well above 90% accuracy.

TUC_MI run1 The first submission was based on a weighted voting between five models which were all trained on the noisy dataset. These five models consisted of one ResNet50, an Inception v3 and three DenseNet201. The weight of the vote

was calculated using the validation accuracy of each model. We trained each of the DenseNets with a different class weight configuration.

TUC_MI run3 This submission consisted of the predictions of one DenseNet201. This model had the best single net validation accuracy of all our trained models. It was only trained on the noisy set and is the last iteration of the fine-tuning process described in 4.1.

TUC_MI run4 Another single DenseNet201 was used for the fourth run, taken from the last iteration of the class weight fine-tuning. Unlike *run3*, this model was mainly trained on the clean data set.

TUC_MI run5 Weighted voting of three models -two DenseNet201 and a single ResNet50 - resulted in our best scores. The two DenseNets were taken from *run3* and *run4*. The ResNet50 model was trained on the noisy set, but with no further weight configurations.

Table 2: The models used in our submission and their respective scores on the official two testset splits.

Submission	Models	Top1-experts	Top1-testset
TUC_MI run1	1 ResNet50, 1 Inception v3, 3 DenseNet201	64%	75.5%
TUC_MI run3	1 DenseNet201	61.3%	71.8%
TUC_MI run4	1 DenseNet201	58.7%	69.8%
TUC_MI run5	1 ResNet50, 2 DenseNet201	64%	77%

6 Conclusion

In our experiments, the DenseNet outperformed Inception v3 and the ResNet50 architectures. The adjustment of the class weights used during training did improve the performance of the DenseNet gradually. A single fine-tuned DenseNet already scored a Top1-accuracy of 71.8% which should increase further with more training time. Considering the strong results of our experiments and the submissions of other participants, we can conclude that machine plant classification is within reach of human-like performance. Some experts are able to identify plants based on images with very high accuracy. Most likely, further improvements of deep learning techniques will close this gap in the next few years.

Our system was implemented with Keras [8] and the source code is publicly available at: https://github.com/Josef-Haupt/ExpertCLEF2018.

References

1. Goëau, H., Bonnet, P., Joly, A.: Overview of ExpertLifeCLEF 2018: how far automated identification systems are from the best experts?. In: CLEF working notes 2018 (2018)
2. Joly, A., Goëau, H., Botelle, C., Glotin, H., Bonnet, P., Planqué, R., Vallinga, W.P., Müller, H.: Overview of LifeCLEF 2018: a large-scale evaluation of species identification and recommendation algorithms in the era of AI. In: Proceedings of CLEF 2018 (2018)
3. Goëau, H., Bonnet, P. and Joly, A.: Plant identification based on noisy web data: the amazing performance of deep learning (LifeCLEF 2017) (2017)
4. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Incpetion Archtecture of Computer Vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818-2826 (2015)
5. He, K., Thang, X., Ren, S., Sun J.: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778 (2015)
6. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K. Q.: Densely Connected Convolutional Networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Vol. 1, No. 2, p. 3 (2016)
7. Lasseck, M.: Image-based Plant Species Identification with Deep Convolutional Neural Networks. In: CLEF working notes 2017 (2017)
8. Chollet, F., et al., https://keras.io (2015)