

Deep Learning based disease diagnostics in agriculture using Convolutional Neural Network and Transfer Learning

Practical Course Project

in the study course Applied Computer Science at the Ravensburg-Weingarten
University of Applied Sciences

Author:

Julia Mueller
Brahmsstrasse 5/1
88085 Langenargen

Tutor:

Mark Schutera
Ravensburg-Weingarten
University of Applied Sciences

Processing period: 01. May 2020 to 10. June 2020

Submission date: 11. June 2020

Abstract

Convolution Neural Networks can be favourably used for the plant disease diagnosis in agriculture due to their ability to extract distinguishing features by itself in case of poorly explicitly describable characteristics of classes. In combination with transfer learning, they are an efficient method with a high classification performance. A pre-trained Xception model has been enhanced by a GlobalAveragePooling layer, Dropout layer for regularization and an output layer for plant diseases classification. A labelled image data set provided by the public data platform Kaggle was used to train the network. The systematic optimization of the classification performance was achieved by fine-tuning the network, by stepwise modification of the model's hyperparameters, by preprocessing of the image data and by variations in the training process. It was shown that especially the complete training of all Xceptions layers in the learning process led to a significant improvement of the classification performance. Further improvements were achieved by reducing the learning rate, introducing Label Smoothing, balancing underrepresented classes in the training data and using the brightness range as data augmentation parameter. Two model settings with detection rates of ~ 94% were found, with one variant having higher accuracy with lower generalization performance and the other variant having slightly lower accuracy with a higher generalization performance.

Task description for the practical course project

The topic of this project was specified in the Kaggle Competition "Plant Pathology 2020 -FGVC7". The corresponding training dataset and test dataset was provided by Kaggle.

1.1 Problem Statement

„Misdiagnosis of the many diseases impacting agricultural crops can lead to misuse of chemicals leading to the emergence of resistant pathogen strains, increased input costs, and more outbreaks with significant economic loss and environmental impacts. Current disease diagnosis based on human scouting is time-consuming and expensive, and although computer-vision based models have the promise to increase efficiency, the great variance in symptoms due to age of infected tissues, genetic variations, and light conditions within trees decreases the accuracy of detection.“ (Kaggle, 2020)

1.2 Specific Objectives

„Objectives of 'Plant Pathology Challenge' are to train a model using images of training dataset to

- 1) Accurately classify a given image from testing dataset into different diseased category or a healthy leaf;
- 2) Accurately distinguish between many diseases, sometimes more than one on a single leaf;
- 3) Deal with rare classes and novel symptoms;
- 4) Address depth perception—angle, light, shade, physiological age of the leaf; and
- 5) Incorporate expert knowledge in identification, annotation, quantification, and guiding computer vision to search for relevant features during learning.“ (Kaggle, 2020)

1.3 Data Description

„Given a photo of an apple leaf, can you accurately assess its health? This competition will challenge you to distinguish between leaves which are healthy, those which are infected with apple rust, those that have apple scab, and those with more than one disease“ (Kaggle, 2020)

Content

Abstract	2
Task description for the practical course project	2
1.1 Problem Statement	2
1.2 Specific Objectives	2
1.3 Data Description	2
1 Introduction	4
1.1 General approaches to machine image classification	4
1.2 Deep Learning Algorithms	4
1.3 Convolutional Neural Network	4
1.4 Transfer Learning	4
2 Convolutional neural network for plant diseases classification	4
2.1 Methodical approach	4
2.2 Structure of the neural network	5
2.2.1 Input	5
2.2.2 Xception	5
2.2.3 GlobalAveragePooling Layer	6
2.2.4 Dropout Layer	6
2.2.5 Output Layer	6
3 Performed experiments	6
3.1 Training data preprocessing	6
3.2 Model fine-tuning	7
3.3 Hyperparameter tuning	7
3.4 Training optimization	7
4 Experimental results	7
4.1 The best results on the public data subset	7
4.2 The best results on the final private data subset	8
5 Discussion	9
5.1 Impact of Xception model fine-tuning	9
5.2 Impact of hyperparameter tuning	9
5.3 Impact of data preprosessing	9
5.4 Impact of training optimization	10
5.5 Discussion summary and approaches for further performance improvements	10
References	11
Declaration of honour	12

1 Introduction

Automatic object classification is the automatic assignment of objects to defined categories/classes on the basis of features. It represents an essential task in many different areas of application in society. The application for plant diseases diagnosis in agriculture, it is the basis for the targeted use of chemicals with the lowest possible impact on nature at the lowest possible economic cost.

1.1 General approaches to machine image classification

There are three basic approaches in the area of machine image recognition / image classification: the rule-based systems, machine learning and deep learning (fig. 1). A detailed discussion of these approaches can be found in (Dimitri Groß, 3 Jan. 2017) and (Rinu Boney Oct 18, 2015)

Contrary to rule-based systems and machine learning, in deep learning the system "does not know the representative features of an object, but learns to extract these features itself progressively. Over time, such systems develop a robust model and are able to recognize any objects independently". (Dimitri Groß, 3 Jan. 2017)

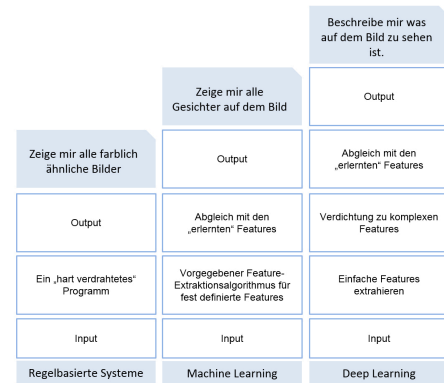


Fig. 1: Rule-Based Systems vs. Machine Learning vs. Deep Learning, Source: Groß, Jan. 2017

1.2 Deep Learning Algorithms

Classification of plant diseases is based on very fine differences in features which are very difficult to describe explicitly. Especially for such problems deep learning methods can favourably be used due to their ability to extract distinguishing features by itself.

The most popular deep learning algorithms are Convolutional Neural Network (CNN), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), Stacked Auto-Encoders, Deep Boltzmann Machine (DBM), Deep Belief Networks (DBN) (J. Brownlee, 5 Dec. 2019). CNN is the algorithm of Deep Learning, which is mainly used for the analysis and processing of image or audio data

1.3 Convolutional Neural Network

„A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. [...] A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers.“ (Margaret Rouse, 2018 April)

A Convolutional Neural Network uses its filters to recognize structures in the input data regardless of location. On the first level, the filters are activated by simple structures such as lines, edges and spots of color. The type of filter is not predetermined, but is learned from the network. In the next level, structures are learned that consist of a combination of these basic structures, e.g. curves, simple shapes, etc.

With each filter level the abstraction level of the mesh increases. Which abstractions finally lead to the activation of the rear layers is determined by the characteristic features of the given classes to be recognized. (Roland Becker, 2019 February)

1.4 Transfer Learning

The data set provided by Kaggle contains only 1821 images. One of the techniques used on CNN when only a very small data set is available is Transfer Learning. One of the techniques for achieving a high classification rate used on CNN when only a very small data set is available is transfer learning. Transfer Learning means using a network which is already pre-trained on an extremely large database of several million images (e.g. ImageNet) for a new task. Therefore a new small task-specific data set may be sufficient to achieve very good results. In transfer learning, the existing knowledge of the pre-trained model is used for the classification of new objects. Depending on the task, it can be decided how many layers will be trained with the new images.

2 Convolutional neural network for plant diseases classification

2.1 Methodical approach

A convolutional neural network with the pre-trained model Xception was chosen as method for classification of different plant diseases and healthy plants, because of the following reasons:

- Convolutional Neural Networks (CNN) are particularly suitable for the machine processing of image or audio data.
- The task description requires special attention to depth perception-angle, light, shade, physiological age of the leaf.

- The different diseases are particularly evident in the different colouring of the leaves and less in their shape. The Xception model gives the colour values a special importance (see also Depthwise Separable Convolution).
- In the Separable Convolutional Layers the deep component is separated from the spatial component. Therefore less parameters are needed. This makes it easier for the net to learn and allows a deep net structure where very complex properties such as the difference between several leaf diseases could be extracted/learned.

2.2 Structure of the neural network

The neural network developed to classify the health status of apple leaves consists of the components shown in Fig. 2:

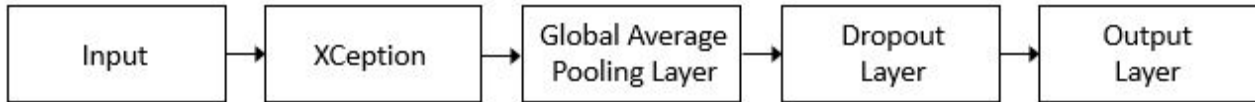


Fig. 2 Architecture of CNN for plant diseases classification

2.2.1 Input

The input of the neural network are batches of tensor image data from the Kaggle training dataset with real-time data augmentation generated with the ImageDataGenerator. The exact parameters for ImageDataGenerator are described in chapter 3.1. The labels for the images are implemented as one-hot encoded labels, which is very well suited for a classification task.

2.2.2 Xception

The heart of the model is the pre-trained Xception model. The characteristic features of this model are:

- „fundamental hypothesis: mapping of cross-channels correlations and spatial correlations can be entirely decoupled.
- composed of 36 convolutional layers forming the feature extraction base of the network
- structured into 14 modules, all of which have linear residual connections around them, except for the first and last modules.“ (S. Osterburg, 9 May 2019)

Figure 4 visualizes the architecture of the Xception model

CNN architecture of Xception based entirely on depthwise separable convolution layers.

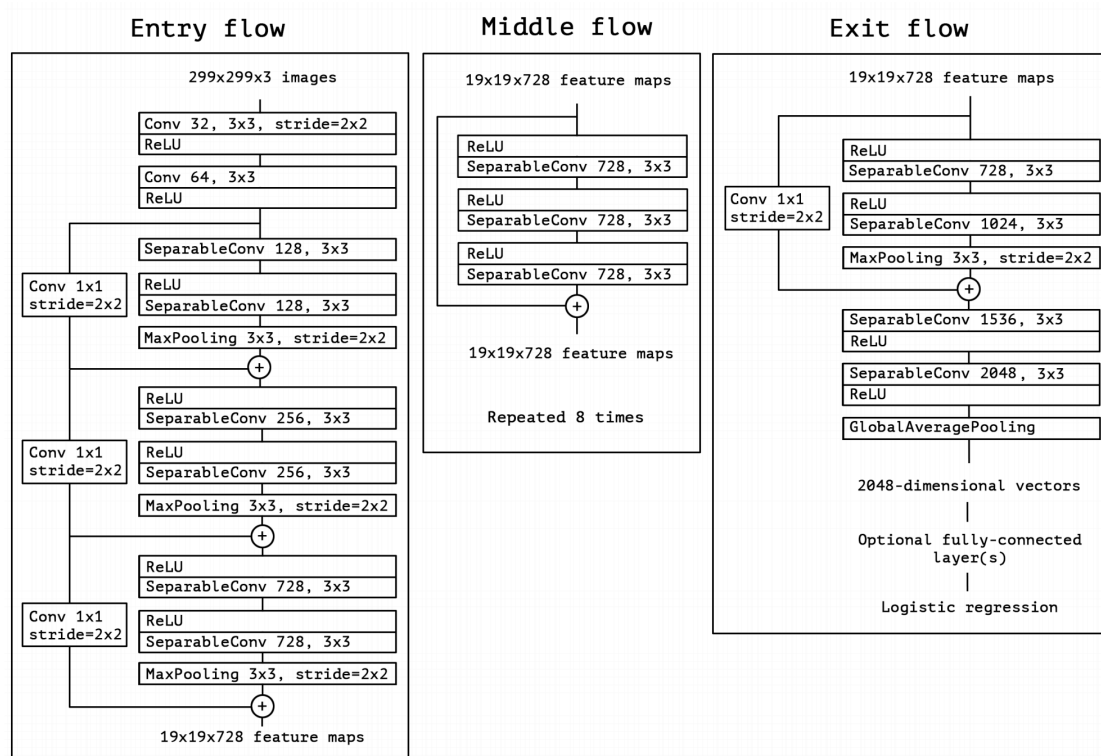


Fig. 3: Xception architecture, Source: S. Osterburg, Xception Architectural Design, 9 May 2019

In Xception, Pointwise Convolution and Depthwise Convolution are separated and implemented in a Separable Convolution Layer.

„In depthwise convolution, we use each filter channel only at one input channel. In the example, we have 3 channel filter and 3 channel image. What we do is — break the filter and image into three different channels and then convolve the corresponding image with corresponding channel and then stack them back“ (Atul Pandey, Sep 9, 2018)

„The pointwise convolution is so named because it uses a 1x1 kernel, or a kernel that iterates through every single point. This kernel has a depth of however many channels the input image has“ (Chi-Feng Wang, August 14, 2018)

„In the separable convolution, we only really **transform the image once** - in the depthwise convolution. Then, we take the transformed image and **simply elongate it to 256 channels**. Without having to transform the image over and over again, we can save up on computational power.“(Chi-Feng Wang, August 14, 2018). This property of the Separable Convolutional Layer allows a deeper network structure.

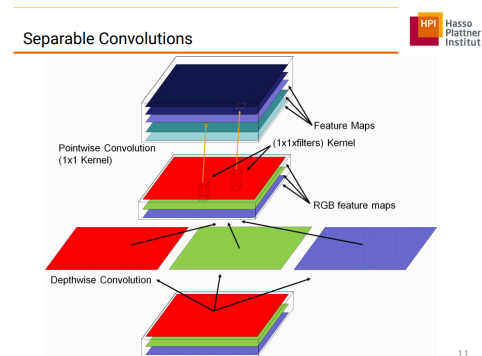


Fig. 5: Separable Convolution, Source: OpenHPI, Exkurs - Exception, Aufbau eines aktuellen CNNs, S.11

2.2.3 GlobalAveragePooling Layer

The connection layer between the output of Xception Model and the next layers is the GlobalAveragePooling Layer (GAP). GAP layer calculates the average output of each feature map in the previous layer. GAP layer reduces each height × width feature map to a single number by simply taking the average of all height and width values. In this way the GAP reduces the number of trainable parameters significantly (e.g. from 7x7x2048 to 1x1x2048). As a result, the model is better prevented by overfitting. Furthermore this increases the speed of the training and prepares the model for the final classification layer. (Adventuresinmachinelearning, 23 May 2020)

2.2.4 Dropout Layer

An additional Dropout layer was introduced between the GAP and the output layer. „The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.“ (TensorFlow Documentation) In the developed model the rate is set to 0.5. This layer has shown a good regularizing effect.

2.2.5 Output Layer

The Output layer is a Dense layer, which has the Softmax function as activation function. Softmax was chosen as activation function, because „Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would. [...] The Softmax layer must have the same number of nodes as the output layer.“ (Machine Learning Crash Course, Jun 2020) The network should divide the input data into 4 classes. Therefore the Output layer has 4 nodes.

3 Performed experiments

For systematization the variety of all settings to improve the performance of the model, the settings and parameters were divided into four categories. Optimizing the settings in one category has formed one "work block".

- In the first block it was tried to optimize the training data preprocessing (see 3.1).
- In the second block it was analyzed which changes in the model parameters lead to the better results (see 3.2).
- In the third block was experimented with the hyperparameters (see 3.3).
- In the fourth block, it has been tried to find the optimal conditions for training execution (see 3.4)

While the parameters of one block were changed, the parameters of all other blocks remained unchanged. After the best combination for one block was found, it was experimented with the parameters of the next block. This procedure was applied several times over all blocks in sequence, with the aim of finally finding the best combination of all parameters. Below are the settings that were tested within each block. As metrics, I used Accuracy. The quality was evaluated according to validation accuracy and the difference between training and validation accuracy (as a measure of generalization).

3.1 Training data preprocessing

The following settings were analyzed:

Tab. 1.a + 1.b analyzed preprocessing settings

Settings	from	to
Splitting into training data/ validation data	0.7/0.3	0.8/0.2
width_shift_range	0.15	0.3
height_shift_range		
zoom_range		
brightness_range	[0.4, 0.9]	[0.5, 1.2]

Settings	from	to
class_weight	no	{ 0: 0.75, 1: 1.75, 2: 0.75, 3: 0.7 }
additional noise	no	yes
Increase by resampling the number of images from the underrepresented class "multiple_diseases"	91	713

3.2 Model fine-tuning

For the Xception model it was experimented with the number of trainable layers. First all Xception layers was set to not-trainable. After that the exit-flow part, then the middle-flow part and finally all Xception layers were set as trainable.

3.3 Hyperparameter tuning

Tab.2 Hyperparameter tuning

Hyperparameter	Tested values		
Optimizer	SGD	RMSProp	ADAM
Learning rate	0,001	0.0001	0.00008
Activation function from output layer	sigmoid	softmax	
Batch size	20	34	48
Loss function	CategoricalCrossentropy without label_smoothing	CategoricalCrossentropy with label_smoothing=0.01	CategoricalCrossentropy with label_smoothing=0.1

3.4 Training optimization

While executing the training it was tried to find out the optimal number of epochs. It was trained in 25, 30, 35 and 40 epochs.

4 Experimental results

The trained models were tested on the test data of Kaggle. The submitted result was evaluated by Kaggle on two different subsets of the test data: a public data subset during the competition and a final private data subset. For the different data subsets, two different models have shown the best results of ~ 94%. One variant has a higher accuracy with lower generalization performance and the other variant shows slightly lower accuracy with a higher generalization performance.

4.1 The best results on the public data subset

The following settings were applied to this model. The differences between this two models are framed in red:

Tab.3 Setting of model with the best public score

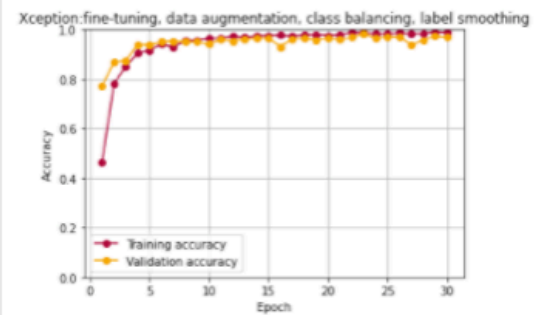
Data preprozessing		Hyperparameter	
Splitting into training data/ validation data	0.8/0.2	Optimizer	ADAM
width_shift_range, height_shift_range, zoom_range	0.15	Learning rate	0.0001
brightness_range	[0.5, 1.2]	Activation function from output layer	softmax
class_weight	no	Batch size	32
additional noise	yes	Loss function	CategoricalCrossentropy with label_smoothing=0.01
Increase by resampling the number of images from the underrepresented class "multiple_diseases"	713		
Model parameter			
Number of trainable Xception layers	all		
Execution of the training			
Number of epochs	30		

Fig.6 Comparison of training and validation accuracy

For the public data subset, the best result reached the accuracy value of 0.94171.

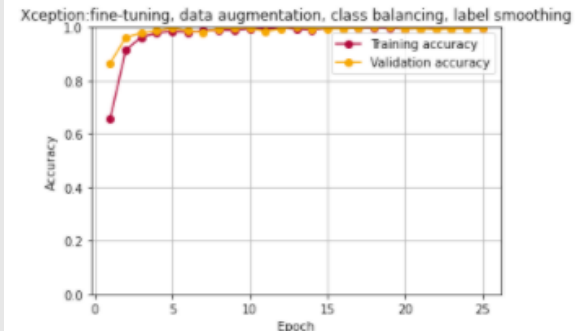
Submission and Description	Private Score	Public Score	Use for Final Score
prediction_adam Js_cb_noise_brightn_range_last.csv 14 days ago by ADL14JM Message: ADL14JM 4	0.91331	0.94171	✓

Fig. 7 The best public score of the network in Kaggle Competition, Source: Kaggle, Plant Pathologie 2020 -FGVC7, May 2020

4.2 The best results on the final private data subset

The following settings were made for this model. The differences to the model described above are framed in red.

Tab.4 Setting of model with the best private score

Data preprocessing		Hyperparameter	
Splitting into training data/ validation data	0.8/0.2	Optimizer	ADAM
width_shift_range, height_shift_range, zoom_range	0.15	Learning rate	0.0001
brightness_range	[0.5, 1.0]	Activation function from output layer	softmax
class_weight	no	Batch size	32
additional noise	no	Loss function	CategoricalCrossentropy with label_smoothing=0.01
Increase by resampling the number of images from the underrepresented class "multiple_diseases"	713		
Model parameter			
Number of trainable Xception layers	all		
Execution of the training			
Number of epochs	25	Fig.8 Comparison of training and validation accuracy	

For the private data subset, the best result was an accuracy value of 0.94085. Although this value is slightly lower than the public score of the model described above, it has only a 1% difference between the public and private score. The model shows much more stable results and therefore the better general performance.

Submission and Description	Private Score	Public Score	Use for Final Score
submission_jm.csv 15 days ago by ADL14JM Message: ADL14JM 03	0.94085	0.93053	<input type="checkbox"/>

Fig. 9 The best private score of my model in Kaggle Competition, Source: Kaggle, Plant Pathologie 2020 -FGVC7, May 2020

5 Discussion

The performed experiments have shown how different settings and parameters impact on accuracy and generalization capability of the network.

5.1 Impact of Xception model fine-tuning

The highest accuracy improvement was achieved when all layers of the Xception model completely were involved in the learning process. The validation accuracy of the classification increased from around 0.5 up to 0.9. In case of very fine differences between the classes the already pre-trained features are not able to achieve sufficient classification results.

5.2 Impact of hyperparameter tuning

Despite the increased accuracy through fine-tuning of Xception Model, the overfitting effect was still very strong. The strong network learned the relatively small data set of training data "by heart" quite fast. While the training accuracy increased to almost 1 after 15 training epochs, the validation accuracy remained constant at around 0.92.

Two main hyperparameters have contributed to the avoidance of overfitting and the achievement of a good generalization. The best regularization effect was achieved by reducing the learning rate to 0.00008 and increasing the batch size from 20 to 32. As a result, the net learned more slowly, but it was able to generalize better the learned knowledge. Comparing the ADAM optimizer with SGD and RMSprop, the ADAM optimizer has shown a slightly better accuracy.

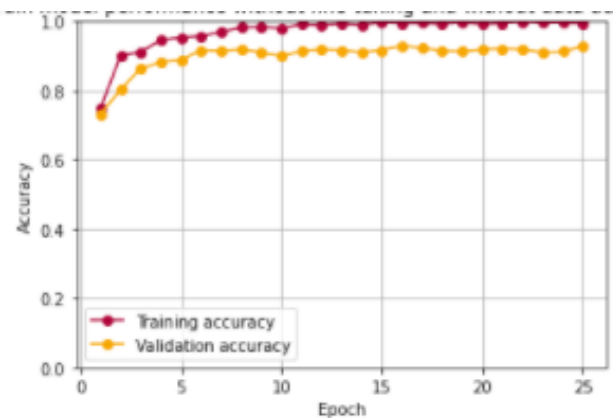


Fig. 10: Training and validation accuracy of model with RMSprop Optimizer, learning rate = 0.001, batch size = 20

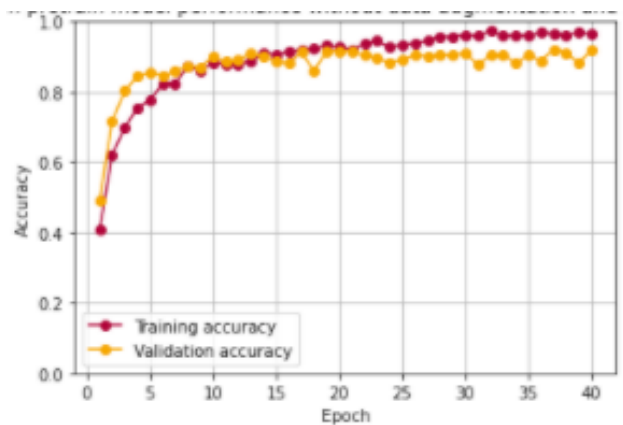


Fig. 11: Training and validation accuracy of model with ADAM Optimizer, learning rate = 0.00008, batch size = 32

An additional increase in accuracy has been achieved by label smoothing using CategoricalCrossentropy Loss function. The label smoothing reduces the one-hot encoded labels by a given factor and thus protects the network from overconfidence. The factor 0.01 has led to the best results.

Finally, overfitting was avoided and a better generalization was achieved by adjusting the hyper parameters.

5.3 Impact of data preprocessing

From the beginning the network was trained with augmented data. Especially in case of a rather small data set, the use of the data augmentation method is very important. This means that the number of images in the training data set has to be increased by modified images. Then the network does not learn the images quickly "by heart" and can better generalize.

Parameters like rotation, shift or zoom did not play a major role. First, the apple leaves on the test and on the training data were already photographed in different orientations and positions on the image. Second, the alignments have nothing to do with the symptoms of the disease and rather contribute to generally improved object recognition.

Brightness has been identified as one of the most important augmentation parameters. Training with images of very different lighting conditions is very important for the diagnosis of individual diseases. E.g. in case of an insufficient training, dark spots on the leaves caused by disease could be mistaken for a shadow. The use of a wide range of brightness [0.5 - 1.0/1.2] on the Kaggle test data has increased the accuracy by almost 0.1.

The use of noise to augment the data is controversial. The main difference between the two models described in chapter 4 (Results) resulting from the use of this parameter. In the first model, the use of noise resulted in good performance on the Kaggle public data subset (accuracy 0.942), but the results were significantly worse when testing on the private subset (0.913). The absence of noisy images in the second model led to more stable results (0.93 in the public score, 0.94 in the private score). The use of noisy images in training complicates the learning for the network and avoids overfitting. However, it is also possible that the learned features may differ too much from the real not noisy test data and cannot be recognized.

With regard to the input data, another feature has proven to be important to achieve a high classification performance. One of the classes (multiple_disease) was strongly underrepresented. While the number of images in the other classes varied in the range of 516 to 622, the multiple_disease class was only represented with 91 images. To balance the number of images the resample module of sklearn.utils was used. Resampling is a technique for replication the data of the underrepresented class according to the random principle. This technique has increased the accuracy on the Kaggle

test data by about 0.2. A further improvement is promised by using the SMOTE (Synthetic Minority Over-sampling Technique) to increase the number of images in underrepresented classes. While resampling simply replicates the images, the SMOTE technique generates new images from existing images, which increases not only the number but also the variety of training images.

5.4 Impact of training optimization

In terms of the number of training epochs carried out, training to 25 to 30 epochs has proven to be sufficient. Longer training did not improve the accuracy. In most cases, after 25 epochs, the network showed constant results (no significant change in both training and validation accuracy).

5.5 Discussion summary and approaches for further performance improvements

The developed classification method based on a convolutional neural network achieves a classification performance of ~94%. With pre-trained models, such as the used Xception model, with frozen weights only a very limited classification accuracy could be achieved because of the very fine feature differences in this specific task. Consequently, the greatest improvement was achieved by fully integrating all layers of the CNN in the learning process. In addition, the importance of a balanced data set in the training became evident. Furthermore important for the generalization performance of the network is a large variance of the training data, in this case especially the brightness variance of the images. Finally, a moderate learning rate and a high batch size are important for a good regularization and to avoid overfitting.

In this project the manual search of the optimal set of parameters and settings was applied. For the further work it can be tried to automate the search for these parameters. The application of Grid Search and Randomized Search are well suited for this purpose. Further optimization methods include Bayesian optimization and Genetic/Evolutionary Algorithms.

There is also great potential for increasing accuracy through better data pre-processing, e.g. the application of the SMOTE technique to compensate for under-represented data. The increase of the data set are seen as a promising approaches to further improvement of the classification method with regard to an increase of the classification and generalization performance. The problem of the small training dataset could be solved e.g. by using the Generative Adversarial Networks.

References

Kaggle (June 6, 2020), Plant Pathologie 2020 -FGVC7, <https://www.kaggle.com/c/plant-pathology-2020-fgvc7>

Dimitri Groß (3 Jan. 2017), Maschinelle Bilderkennung mit Big Data und Deep Learning, <https://jaxenter.de/big-data-bildanalyse-50313>

Rinu Boney (Oct 18, 2015), Theoretical Motivations for Deep Learning, <https://rinuboney.github.io/2015/10/18/theoretical-motivations-deep-learning.html>

J. Brownlee (5 Dec. 2019), A Tour of Machine Learning Algorithms, <https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/>

Margaret Rouse (April, 2018), Convolutional neural network, <https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network>

Roland Becker, (February 6, 2019) Convolutional Neural Networks – Aufbau, Funktion und Anwendungsgebiete, <https://jaai.de/convolutional-neural-networks-cnn-aufbau-funktion-und-anwendungsgebiete-1691/#:~:text=In%20einem%20Convolutional%20Neural%20Network,sind%201%3A1%20erhalten%20bleiben.>

OpenHPI (April, 2020), Exkurs - Xception, Aufbau eines aktuellen CNNs

Chi-Feng Wang (August 14, 2018), A Basic Introduction to Separable Convolutions, <https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728>

Atul Pandey (Sep 9, 2018), Depth-wise Convolution and Depth-wise Separable Convolution, <https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec>

S. Osterburg (9 May 2019), Xception Architectural Design, <https://stephan-osterburg.gitbook.io/coding/coding/ml-dl/tensorflow/ch3-xception/xception-architectural-design>

Adventuresinmachinelearning (23 May 2020), An introduction to Global Average Pooling in convolutional neural networks, <https://adventuresinmachinelearning.com/global-average-pooling-convolutional-neural-networks/>

TensorFlow Documentation, https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

Machine Learning Crash Course (Jun 2020), <https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax>

Declaration of honour

I assure that I have written the enclosed project with the topic

"Deep Learning based disease diagnostics in agriculture using Convolutional Neural Network and Transfer Learning"

myself, that I have not used any other sources and aids than those indicated, and that I have marked all parts of the work that have been taken over literally or correspondingly.

A handwritten signature in black ink, appearing to be 'H. Khan', is centered on the page.