remote sensing

MDPI

Letter

Application of Deep Learning Architectures for Accurate Detection of Olive Tree Flowering Phenophase

Mario Milicevic *, Krunoslav Zubrinic, Ivan Grbavac and Ines Obradovic

Department of Electrical Engineering and Computing, University of Dubrovnik, 20000 Dubrovnik, Croatia; krunoslav.zubrinic@unidu.hr (K.Z.); ivan.grbavac@unidu.hr (I.G.); ines.obradovic@unidu.hr (I.O.)

* Correspondence: mario.milicevic@unidu.hr

Received: 31 May 2020; Accepted: 29 June 2020; Published: 2 July 2020

Abstract: The importance of monitoring and modelling the impact of climate change on crop phenology in a given ecosystem is ever-growing. For example, these procedures are useful when planning various processes that are important for plant protection. In order to proactively monitor the olive (*Olea europaea*)'s phenological response to changing environmental conditions, it is proposed to monitor the olive orchard with moving or stationary cameras, and to apply deep learning algorithms to track the timing of particular phenophases. The experiment conducted for this research showed that hardly perceivable transitions in phenophases can be accurately observed and detected, which is a presupposition for the effective implementation of integrated pest management (IPM). A number of different architectures and feature extraction approaches were compared. Ultimately, using a custom deep network and data augmentation technique during the deployment phase resulted in a fivefold cross-validation classification accuracy of 0.9720 ± 0.0057. This leads to the conclusion that a relatively simple custom network can prove to be the best solution for a specific problem, compared to more complex and very deep architectures.

Keywords: deep learning; convolutional neural networks; pattern recognition; data augmentation; olea europaea; integrated pest management

1. Introduction

Climate change has had a significant influence on the allocation and population of various species on the planet, including the olive (*Olea europaea L.*) and olive pests [1,2]. The Mediterranean is the main olive-producing area of the world. It is therefore susceptible to increasing challenges in the field of integrated pest management (IPM). This also includes organic farming systems. Their practices are based in ecology, an example of which is biological pest management. Tang et al. [3] state that the optimal timing for pest management is of great importance here, together with taking into account the phenological stages of the treated plants, how dense the starting population of pests is, what their natural enemies' ratios and release timings are, how the insecticides are dosed and when they are applied, as well as what the rates of instantaneous killings of these pesticides are when looking at both the pests and their natural enemies. Therefore, there has been a lot of research interest in modelling a plant's adaptation to its surrounding climate, as well as its response to recent climate changes [4]. Olives are the topic of many such papers and models, for example, De Melo-Abreu et al. [5], Osborne et al. [6] and Aguilera et al. [7]. In [8], it has been concluded that the climate (especially the temperature and photoperiod) greatly influence olive flowering. However, these results differ greatly with varying geographical altitude and latitude. Similarly, Garcia-Mozo et al. [9] state that different olive cultivars have flowering phenology variations, which should also be taken into account.

Additionally, experience has shown that other factors within the same orchard can alter the beginning of the flowering phase up to multiple days [10]. These factors are more difficult to model, such as the age and general condition of trees or the microlocation (e.g., facing the east or having a higher altitude). This led Osborne et al. [6] to conclude that, in order to use any phenological model for a single olive population on the regional level, the model has to first be tested by using a great number of different factors, such as various locations and cultivars.

At the same time, certain pests have adapted to the phenology of olive trees exceptionally well. The olive moth (Prays oleae), for example, develops three generations within the same year [11]. Treatments have to be used against two generations: the anthophagous (flower) generation and the carpophagous generation. At the moment, there are a number of standard control methods for the olive moth, including conventional pesticide applications. However, it is not advised to control the spring generation of moths with insecticides, as this results in catastrophic consequences for the beneficial animal species in olive orchards as well [12]. An alternative and effective method of microbiological control is the application of Bacillus thuringiensis, with its optimal application date being when 5% of the flowers have opened [13]. However, due to the previously explained reasons, in larger olive orchards it is hard to determine the correct point in time for the application of this method. This is why the goal of this study is to test the possibilities of applying machine learning (ML) and, especially, deep learning (DL) technologies in the field of phenological stage classification, with the aim of achieving optimal pest management.

The term deep learning refers to the use of artificial neural network (ANN) architectures consisting of a large number of processing layers. With the development of hardware (especially GPUs), the aforementioned models have become computationally feasible, which resulted in many possible applications. They are used in the fields of image and voice recognition, remote sensing and other complex processes which deal with the analysis of large amounts of data. The use of deep learning techniques for agriculture began a number of years ago, but to a relatively limited degree. Kamilaris and Prenafeta-Boldú [14] offer a detailed overview of concrete issues in agriculture, the used frameworks, models, sources and data preprocessing, as well as the achieved performance data. It can be concluded that, in the context of use in agriculture, deep learning techniques are highly accurate, achieving better results than the techniques currently commonly used in image processing.

This paper is structured as follows: Section 2 provides the basic data on the original dataset used in the experiment, as well as on the tested deep learning algorithms. Section 3 introduces additional improvements made during the deployment phase and includes a discussion of the results reached as part of the experiment. Finally, the main findings are summarized in Section 4.

2. Materials and Methods

2.1. Dataset

The study was carried out in southern Croatia. During April, May and June of 2019, hundreds of images were collected with stationary cameras in an olive orchard. The images were obtained during multiple days and at all times of the day. This means that they were taken in various lighting conditions. The weather conditions included clear, sunny weather to overcast and cloudy weather.

For the initial proof of concept, a Sony 16 MP entry-level mirrorless camera was used. Further on in the experiment, we used a C920 HD Pro webcam, which can take 15 MP photos. Both cameras were set at a distance of 40 to 50 cm from the tree canopy. The format used with both cameras was JPEG. In order to increase the sharpness and reduce potential noise due to lower lighting conditions, the images were resized to 1600×900 pixels.

For the purpose of machine learning, six image patches (256×256 pixels) were extracted from the central part of each original image. This ensured a balanced dataset which consisted of 1400 images, showing details of the tree canopies during various phenological stages. Histogram equalization was applied to each image patch to reduce the effect of varying lighting conditions. Following that,

edge detection algorithms were applied to the images with the goal of finding the most suitable variant for the experiment. All of the mentioned preprocessing steps are shown in Figure 1.

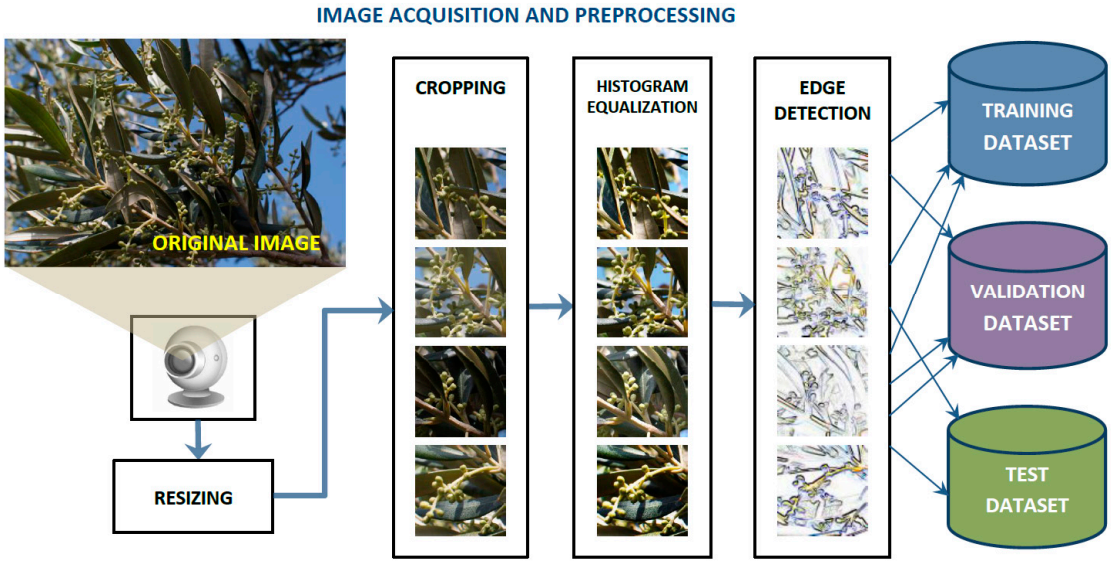

Figure 1. Flow diagram for image preprocessing.

With the aid of an expert in this field, the images were classified as zero or one, depending on whether the start of the flowering phase was observed. Both classes have the same number of samples (700). Figure 2 shows examples of both classes (upper row—class 0, lower row—class 1).

Figure 2. Details of canopies during various phenological stages (upper row—class 0, lower row—class 1).

The dataset was then divided into three parts—200 images were set aside for the model's final testing (test dataset), while the rest (1200 images) were divided into a training dataset of 1000 images and a validation dataset of 200 images. All datasets have the same class ratio as the original dataset. The validation dataset was used during the learning phase to tune the hyperparameters of a classifier (i.e., the architecture, number of epochs, etc.) and then for the final model selection [15]. The test dataset was never used during the training phase, so it could be used for a less biased final estimate of the model's ability to generalize.

As expected during the course of the experiment, the dataset with 1000 images was insufficient to successfully carry out the learning phase. This is why data augmentation was used as the primary method for overfitting reduction in the context of a limited learning dataset [16]. Original images were used as templates to generate new artificial learning examples. For this purpose, various image distortion techniques were used at random, such as zooming, cropping, translating, flipping the image horizontally or vertically, rotation and color modification. This resulted in an increase in the learning dataset to 7000 images.

The initial experiment tested the possibilities of using technology for the detection of the flowering phase onset, i.e., for the recognition of open flowers. According to the Biologische Bundesanstalt, Bundesortamt, Chemiche Industrie (BBCH) scale, this is the transition from phase 59 to phase 60 or 61 [17]. The task is all but trivial as, visually, open flowers vary greatly, depending on the angle and distance of the camera, the lighting (light intensity, color temperature, etc.), objects obstructing the view of the flowers, and other conditions. Figure 3 shows various examples of open flowers. Note that the shown details were magnified and represent around 5% of the size of the images from the learning dataset.

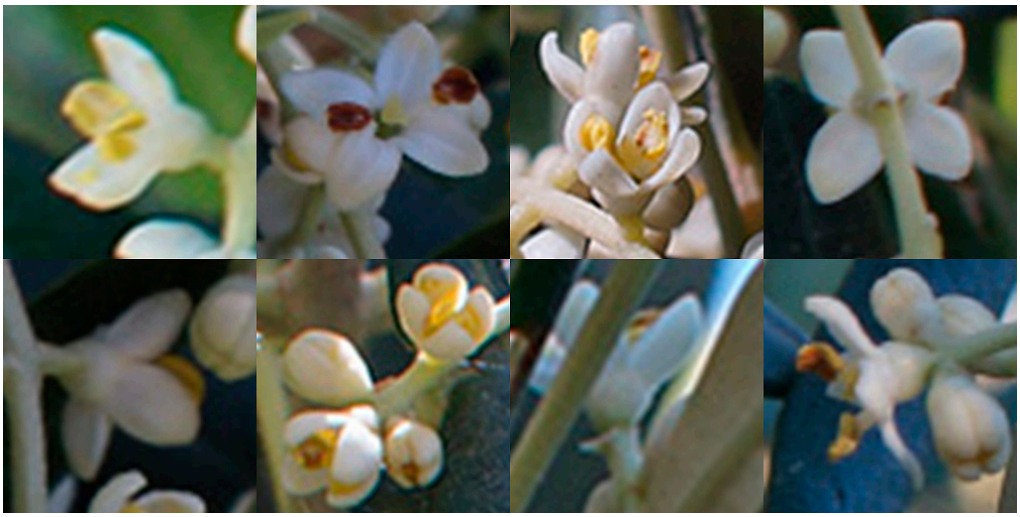

Figure 3. Examples of open flowers (magnified details).

2.2. Experimental Study

Various deep convolutional neural networks (CNNs) were tested, including well-known deep CNNs, such as VGG19 (Visual Geometry Group) [18], InceptionResNetV2 [19], Xception [20] and ResNet50 [21].

Approaches to learning from scratch were also compared to the application of transfer learning [22]. CNN training was implemented with the Python deep-learning framework, including Keras [23] and TensorFlow [24] libraries.

The hardware used to carry this out was an NVidia GeForce GTX 1080 Ti graphics processing unit (GPU) with 11GB memory, AMD Ryzen 5 CPU and 32 GB of RAM. The operating system was Ubuntu 16.04 Linux. We conducted our CNN training on the NVidia CUDA-nabled GeForce GTX 1080 Ti GPU.

As the aforementioned architectures did not reach the expected or satisfactory results, neither for learning speed, nor for classification accuracy, a custom model inspired by a VGG architecture was included in the experiment. The model is shown in Figure 4. A number of convolutional, pooling, normalization and fully connected layer combinations were tested with the goal of reaching the desired results for the test dataset. The research proved that, for this specific problem and amount of data, there is an efficient architecture which is far simpler than the general very deep models.

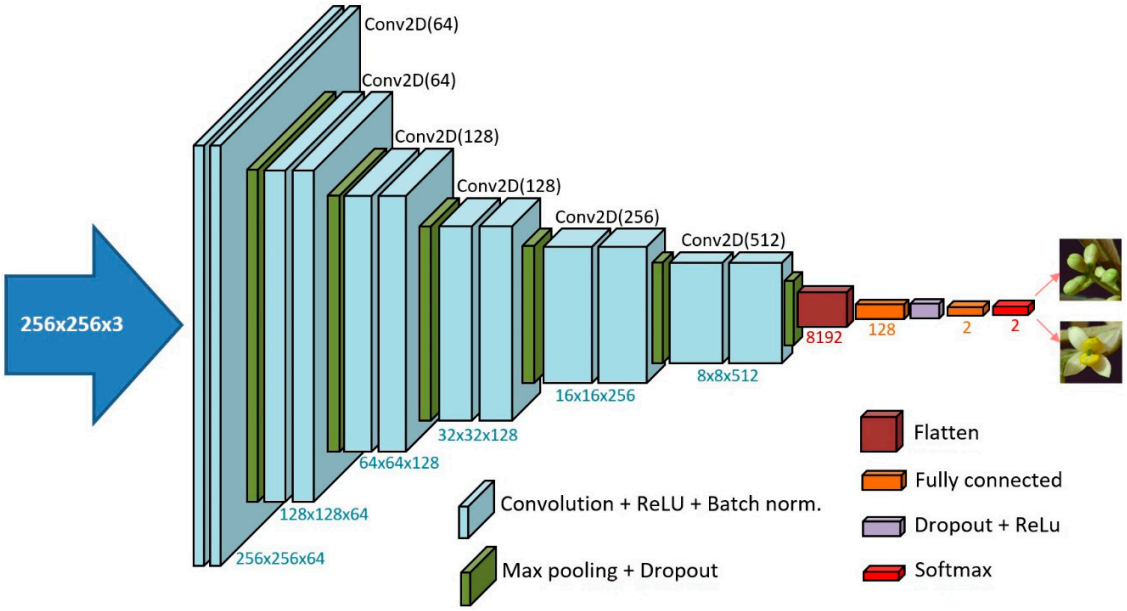

Figure 4. The structure of the custom convolutional neural network (CNN).

The results of the test are shown in Table 1, where the percentage of accuracy reached for the unseen test dataset is displayed. Of course, the training accuracy is significantly better, and is sometimes even close to 100%. However, this is a case of overfitting, i.e., a model's insufficient ability to generalize. This ability to generalize has proven to be practically the most important characteristic of a network for the application intended in this research.

Table 1. Performance comparison of various CNN models (max. 100 epochs).

CNN	Depth	Dataset	Parameters	Accuracy (%)
VGG19	19	Augm.	24,219,106	69.50
VGG19 (pretrained)	19	Augm.	24,219,106	*54.00
InceptionResNetV2	164	Augm.	55,912,674	65.50
InceptionResNetV2 (pretrained)	164	Augm.	55,912,674	66.50
Xception (pretrained)	71	Augm.	22,910,480	67.00
ResNet50	50	Augm.	25,687,938	64.00
ResNet50 (pretrained)	50	Augm.	25,687,938	*54.00
Custom CNN	12	Augm.	3,614,768	91.50
Custom CNN	13	Augm.	3,746,402	93.50
Custom CNN	14	Orig.	6,106,242	88.50
Custom CNN	14	Augm.	6,106,242	94.50
Custom CNN	15	Augm.	6,892,130	93.50
Custom CNN	16	Augm.	10,039,458	91.50

The mentioned state-of-the-art deep architectures were tested by starting the learning process from scratch with random weight initialization, as well as in transfer learning scenarios using ImageNet pretrained models. It is well known that with transfer learning, a significant domain discrepancy results in a significant drop in feature transferability for higher layers. This is why a large number of models with various numbers of pre-trained layers were tested. The results showed that some networks (especially those marked with *) suffer from a convergence problem due to insufficient training data. They also require quite a long time to converge. Tn the context of this issue and in relation to available data, an observation was made that contrasts prior experience: The initialization of the weights from the pretrained network and fine tuning do not significantly contribute to a higher accuracy of classification.

As it is visible in Table 1, the best results were achieved with a variant of the custom (VGG-inspired) deep convolutional network, with the use of an augmented training dataset. It was also found that the optimal architecture includes 14 learnable weights layers. Tested models also included fewer or more layers. The results show architectures with 12, 13, 15 and 16 layers, which all reached somewhat poorer results with the test dataset. This shows that for this specific issue there is an optimal level of network complexity.

Details of the optimal architecture are shown in Table 2, where Conv2D(n) denotes the 2D convolution layer with n filters, ACT denotes the activation function, BN denotes the batch normalization layer, MP denotes the max pooling layer and DR denotes the dropout layer. Tested activation functions used in this model are rectified linear units (ReLU) and leakage-rectified linear units (LReLU) [25].

Table 2. Detailed architecture of proposed custom CNN.

Layer Type	Output Shape	#Param
Input	256 × 256×3	-
Conv2D(64)	256 × 256 × 64	640
ACT(LReLu)-BN	256 × 256 × 64	1,024
Conv2D(64)	256 × 256 × 64	36,928
ACT(LReLu)-BN	256 × 256 × 64	1,024
MP(2,2)+DR(0.1)	128 × 128 × 64	-
Conv2D(64)	128 × 128 × 64	36,928
ACT(LReLu)-BN	128 × 128 × 64	512
Conv2D(64)	128 × 128 × 64	36,928
ACT(LReLu)-BN	128 × 128 × 64	512
MP(2,2)+DR(0.2)	64 × 64 × 64	-
Conv2D(128)	64 × 64 × 128	73,856
ACT(LReLu)-BN	64 × 64 × 128	256
Conv2D(128)	64 × 64 × 128	147,584
ACT(LReLu)-BN	64 × 64 × 128	256
MP(2,2)+DR(0.1)	32 × 32 × 128	-
Conv2D(128)	32 × 32 × 128	147,584
ACT(LReLu)-BN	32 × 32 × 128	128
Conv2D(128)	32 × 32 × 128	147,584
ACT(LReLu)-BN	32 × 32 × 128	128
MP(2,2)+DR(0.2)	16 × 16 × 128	-
Conv2D(256)	16 × 16 × 256	295,168
ACT(LReLu)-BN	16 × 16 × 256	64
Conv2D(256)	16 × 16 × 256	590,080
ACT(LReLu)-BN	16 × 16 × 256	64
MP(2,2)+DR(0.2)	8 × 8 × 256	-
Conv2D(256)	8 × 8 × 512	1,180,160
ACT(LReLu)-BN	8 × 8 × 512	32
Conv2D(256)	8 × 8 × 512	2,359,808
ACT(LReLu)-BN	8 × 8 × 512	32
MP(2,2)+DR(0.2)	4 × 4 × 512	-
Flatten	8192	-
Dense(128)	128	1,048,704
ACT(LReLu)-DR(0.3)	128	-
Dense(2)	2	258
ACT(softmax)	2	-
TOTAL		*6,106,242*

The size and configuration of the applied network is the result of a large number of experiments, including the grid search approach with the goal of determining certain parameters. These are the values of the more important hyperparameters: the number of epochs is 100, mini-batch size is 16, initial learning rate is 0.0001, the optimizer is RMSProp (Root Mean Square Propagation) [26]. During the

learning phase, the callback mechanism was used to save the best model. This assessment was based on the model's performances with the validation dataset. The dynamic learning rate is controlled by the callback function that reduces the learning rate when a defined metric has stopped improving for a given number of epochs.

In order to prevent neural networks from overfitting, batch normalization (BN) was used [27]. This method, at the same time, reduces the internal covariate shift and accelerates the training of deep networks. However, experimentation has shown that, for this specific experiment, it is necessary to use the classic dropout method [28] for additional control, even though using BN practically eliminates the need to use dropout according to literature.

The input was a fixed-size 256 × 256 RGB image, which was processed in various ways during the experiment in order to find effective features. For example, it was attempted to use information about color changes in the flower bud at the beginning of the flowering phase. However, it was concluded that the shape of the flower is actually a more reliable piece of information. This is why multiple methods were used to find the necessary features, especially various edge detection methods, including classic approaches (such as Sobel and Canny methods).

However, the best results were achieved by the newer physics-inspired method—phase stretch transform (PST) [29]. The authors recommend using an algorithm that works through a nonlinear dispersive phase operation. It transforms the image by emulating the propagation of light through a physical medium with a specific warped diffractive property. The authors concluded that the output phase of the transforming reveals transitions in image intensity, which can be used for edge detection and feature extraction. In [30], the authors demonstrated that the PST algorithm, due to its natural equalization mechanism, is able to provide more information on contrast changes in both bright and dark areas of the image. Conventional edge derivative operators fail to visualize the sharp contrast changes in both cases.

This algorithm has been successfully applied to various application areas. For example, in [31], authors have shown that the proposed algorithm exhibits image segmentation performance with an accuracy of 99.74%.

Figure 5 shows the results of PST algorithm use. For the proposed PST method, the designed parameters are LPF = 0.2, phase strength S = 1, warp strength W = 10, minimum threshold $T_{min} = -0.5$ and maximum threshold $T_{max} = 0.5$.

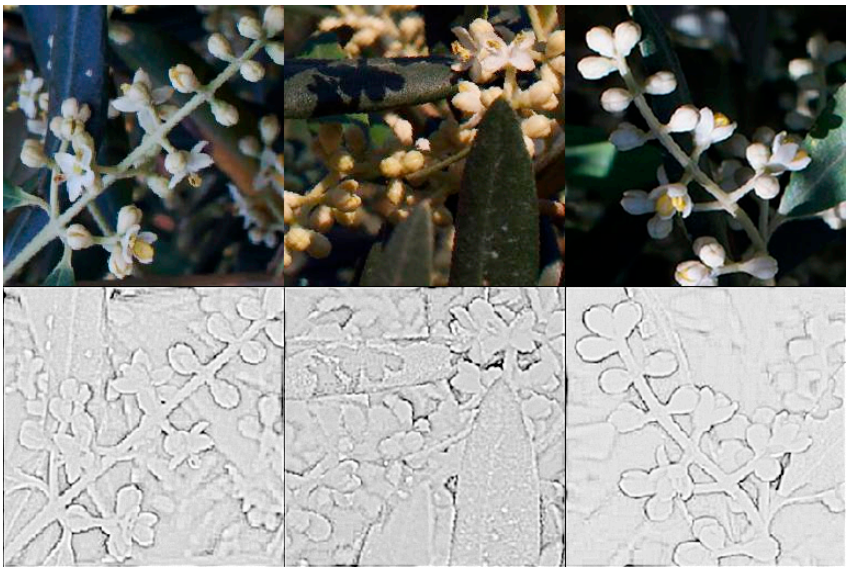

Figure 5. Application of the phase stretch transform (PST) algorithm.

3. Results and Discussion

The experiments showed that the recommended model reaches the best results on the validation dataset after 37 epochs. At that point in time, the validation accuracy is 0.940. Using the same model on the test dataset results in an accuracy of 0.945.

Figure 6 shows details of the experiment realization results which helped find the optimal architecture. First, the impact of the large training dataset size is of note. Even though, using 1000 examples, the model can achieve an accuracy for the training dataset reaching up to 90%, the result of applying this model on the validation dataset (accuracy 60%) shows that this is a case of overfitting. A satisfactory level of generalization could only be reached after increasing the size of the training dataset to 7000 examples. This shows that, considering the limited amount of original data, data augmentation was a necessary step.

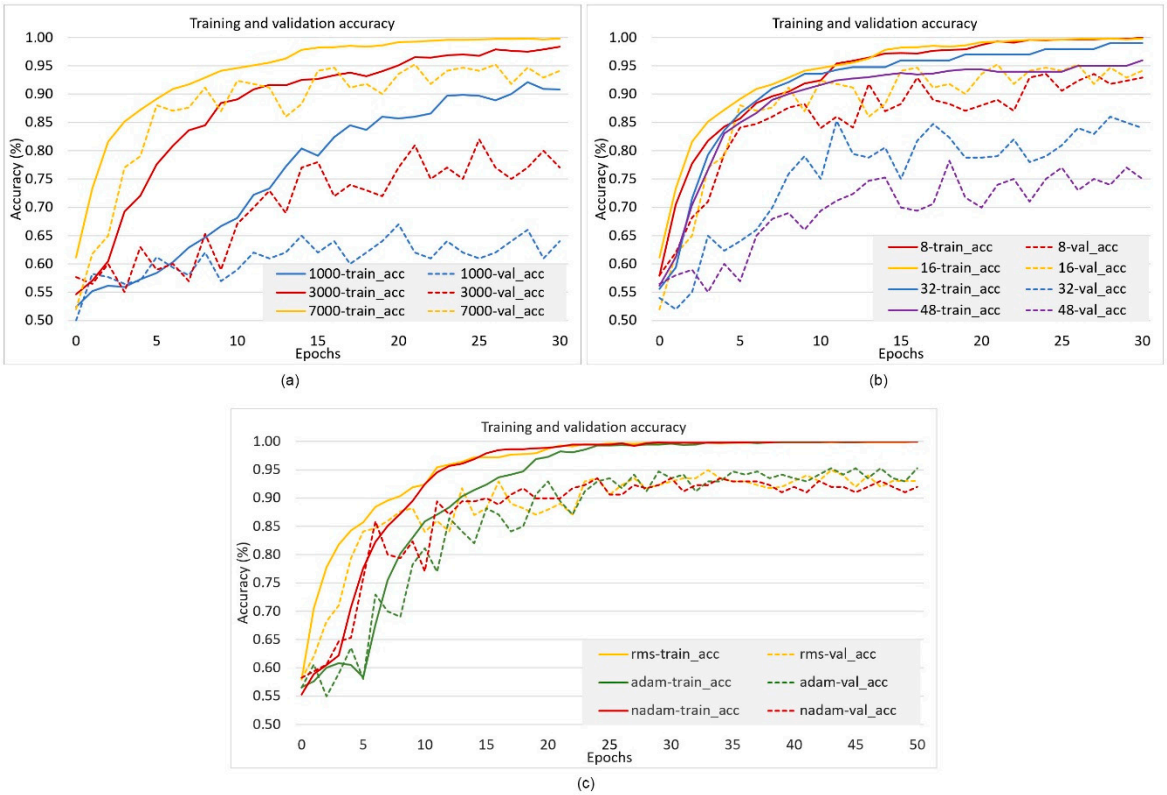

Figure 6. Traces of training and validation accuracy during custom CNN training with (**a**) different training dataset size, (**b**) different batch size and (**c**) different optimizers.

Similarly, in the context of this issue and the available data, it was proven that the best results are achieved with batch sizes of eight and 16. Using a larger batch size can have a negative effect on the accuracy of the network, especially considering a model's ability to generalize. This is seen in the results achieved with the validation dataset. These are, in part, related to the findings in [32]: on the one hand, increasing the batch size reduces the range of learning rates that provide stable convergence and acceptable test performance, which was to be expected. On the other hand, small batch sizes provide more up-to-date gradient calculations, which results in more stable and reliable training.

When it comes to optimizers, all the applied adaptive optimizers (RMSProp, Adam [33], Nadam [34]) achieved similar results after a sufficient number of epochs. However, some differences in the speed of convergence can be noticed. The differences in the respective model's ability to generalize were not significant. A classic SGD optimizer was also tested, but its slow convergence as an issue, so its effect is not comparable to the effect of the applied adaptive optimizers.

Evolution of the accuracies and losses for training and test datasets are shown in Figure 7. The optimal architecture shown in Table 2 was used, applying the hyperparameters obtained by the experiments described in Figure 6.

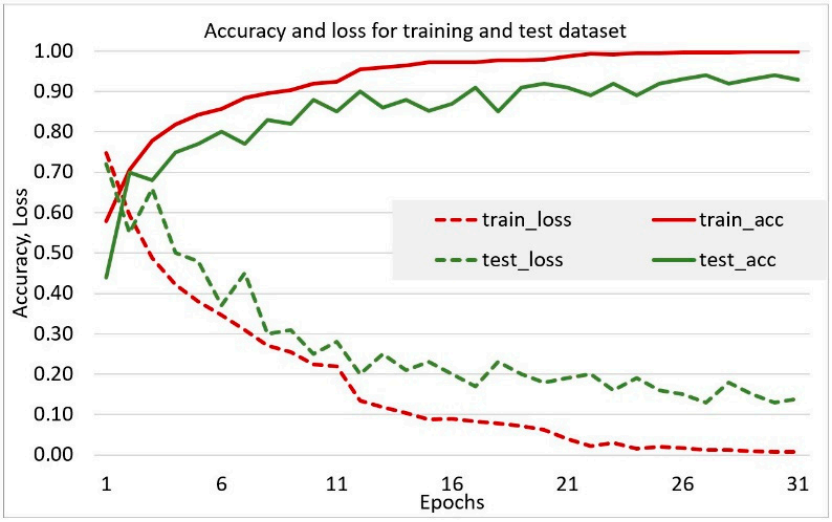

Figure 7. Accuracy and loss for training and test datasets.

Figure 8 shows the confusion matrix, which summarizes the best classifier performance. As already mentioned, the accuracy is 0.9450. The model produced a similar number of false positives and false negatives, so the precision amounts to 0.9406, recall (sensitivity) is 0.9500, and the F1-score (the harmonic mean of the precision and recall) is 0.9453.

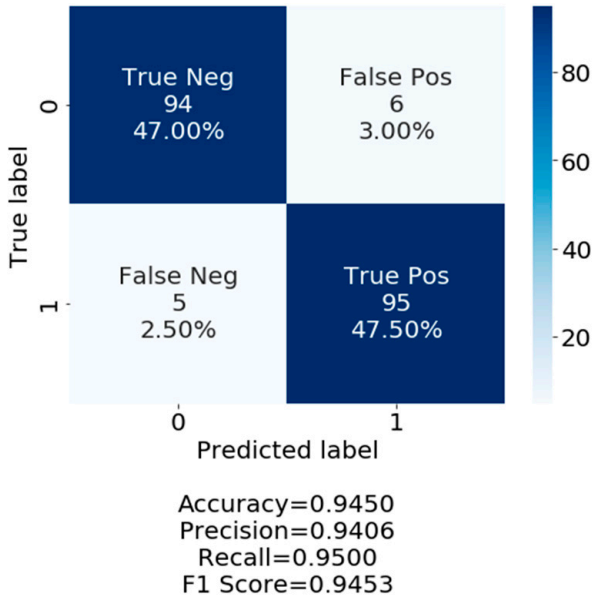

Figure 8. Confusion matrix for the test samples.

Since a number of test samples (5.5%) were still wrongly classified, there is obviously space for the further performance improvement of the model. This is why the experiment shown in Figure 9 was conducted.

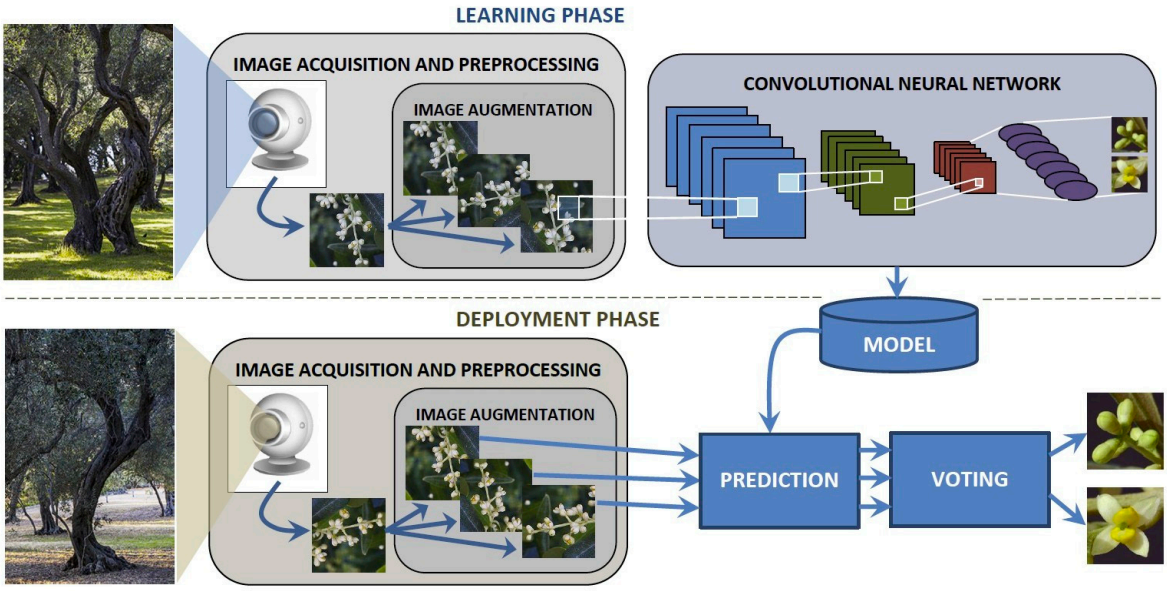

Figure 9. Neural network architecture for phenophases detection.

Our idea is based on the fact that, due to their limited size, the application of an image augmentation technique on learning datasets during the learning phase is of great importance—the random changes applied to training examples can reduce a model's dependency on individual features. This means that this methodology can increase the model's capability for generalization.

This is why it can be concluded that the same technique, used in the final model application phase, could further improve the performance. Therefore, for every image in the test dataset, six new examples were created by using the same data augmentation transformations as in the learning phase.

Figure 10 shows the original test image and six corresponding generated augmented images. The previously saved model was used on each of these artificially created examples, as well as on the original test image (a total of seven images), and the decision about the final classification of the test example was made based on the majority voting principle.

Figure 10. Original test example and six generated augmented images.

Such use of a data augmentation technique during the deployment phase resulted in an improvement in the model's performances by another 3%. This means that the accuracy of the test dataset rose from 0.9450 to 0.9750. Precision amounts to 0.9703, recall (sensitivity) is 0.9800, and the F1-score (the harmonic mean of the precision and recall) is 0.9751.

Finally, the model's robustness was evaluated by a variant of 5-fold stratified cross-validation. Available data was divided fivefold, with each fold having training, validation and test datasets. Validation datasets were used for regularization by the early stopping of learning process. We found only a small difference in the accuracy was reached for the test datasets. The average classification accuracy and standard deviation were 0.9720 ± 0.0057.

To our knowledge, this is the first study that addresses the application of imaging sensors and deep learning techniques on the detection of olive tree flowering phenophases. This area has been the subject of several papers. However, authors used essentially different approaches, and therefore the results cannot be compared directly. For example, the authors in [35] examined the impact of weather-related variables on flowering phenophases and constructed models based on regression. Similarly, the authors in [36] investigated use of artificial neural networks in olive phenology modelling, but again using meteorological data. Aguilera et al. [7] developed pheno-meteorological regression models to forecast the main olive flowering phenological phases.

4. Conclusions

Our study assessed the efficiency of various models based on the CNN architectures in the context of the phenological state classification of plants—in this case, olive trees. The conducted experiment showed that the aforementioned classification can very efficiently be realized by using deep learning algorithms. The best results were reached with a custom VGG-inspired network, which includes 14 learnable weight layers, with the use of an augmented training dataset. The accuracy can be improved further by using data augmentation and majority voting procedures during the deployment phase, where the final classification accuracy amounts to 97.20%. This means that the entire process is viable and applicable under real conditions.

To illustrate the importance of the monitoring of plant phenology, it should be noted that the European Union requires the application of eight principles of IPM that are part of sustainable farm management [37]. Principle 3 is titled "Decision based on monitoring and thresholds", and it assumes "an opportunity to develop a new generation of decision support systems". The suggested approach allows for the optimized timing of chemical or biological crop protection applications. For example, a specific protection product may have an optimal application date when 5% of the flowers have opened. For an individual microlocation, the application time window can be limited to only 2 to 3 days, which requires the precise and timely detection of crop phenophases.

In the future, the proposed system will be applied to classify a greater range of phenological stages, which represent an important parameter when applying various agricultural procedures. We also plan to use images collected by a drone. Similarly, by securing the necessary learning datasets, the same model can be applied to other cultures as well.

Author Contributions: Conceptualization, M.M. and K.Z.; methodology, M.M.; software, M.M., K.Z., I.G. and I.O.; validation, K.Z. and I.G..; formal analysis, M.M.; investigation, M.M. and K.Z.; resources, M.M.; data curation, M.M., K.Z., I.G. and I.O.; writing—original draft preparation, M.M.; writing—review and editing, K.Z.; visualization, I.G. and I.O.; supervision, M.M.; project administration, I.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Moriondo, M.; Bindi, M. Impact of climate change on the phenology of typical Mediterranean crops. *Ital. J. Agrometeorol.* **2007**, *3*, 5–12.
2. Gutierrez, A.P.; Ponti, L.; Cossu, Q.A. Effects of climate warming on olive and olive fly (Bactrocera oleae (Gmelin)) in California and Italy. *Clim. Chang.* **2009**, *95*, 195–217. [CrossRef]
3. Tang, S.; Tang, G.; Cheke, R.A. Optimum timing for integrated pest management: Modelling rates of pesticide application and natural enemy releases. *J. Theor. Biol.* **2010**, *264*, 623–638. [CrossRef] [PubMed]
4. Chuine, I.; Cour, P.; Rousseau, D.D. Selecting models to predict the timing of flowering of temperate trees: Implications for tree phenology modelling. *Plant Cell Environ.* **1999**, *22*, 1–13. [CrossRef]
5. De Melo-Abreu, J.P.; Barranco, D.; Cordeiro, A.M.; Tous, J.; Rogado, B.M.; Villalobos, F.J. Modelling olive flowering date using chilling for dormancy release and thermal time. *Agric. For. Meteorol.* **2004**, *125*, 117–127. [CrossRef]
6. Osborne, C.P.; Chuine, I.; Viner, D.; Woodward, F.I. Olive phenology as a sensitive indicator of future climatic warming in the Mediterranean. *Plant Cell Environ.* **2000**, *23*, 701–710. [CrossRef]
7. Aguilera, F.; Fornaciari, M.; Ruiz-Valenzuela, L.; Galán, C.; Msallem, M.; Dhiab, A.B.; Díaz-de La Guardia, C.; del Mar Trigo, M.; Bonofiglio, T.; Orlandi, F. Phenological models to predict the main flowering phases of olive (Olea europaea L.) along a latitudinal and longitudinal gradient across the Mediterranean region. *Int. J. Biometeorol.* **2015**, *59*, 629–641. [CrossRef]
8. Orlandi, F.; Vazquez, L.M.; Ruga, L.; Bonofiglio, T.; Fornaciari, M.; Garcia-Mozo, H.; Domínguez, E.; Romano, B.; Galan, C. Bioclimatic requirements for olive flowering in two Mediterranean regions located at the same latitude (Andalucia, Spain, and Sicily, Italy). *Ann. Agric. Environ. Med.* **2005**, *12*, 47.
9. Garcia-Mozo, H.; Orlandi, F.; Galan, C.; Fornaciari, M.; Romano, B.; Ruiz, L.; de la Guardia, C.D.; Trigo, M.M.; Chuine, I. Olive flowering phenology variation between different cultivars in Spain and Italy: Modeling analysis. *Theor. Appl. Climatol.* **2009**, *95*, 385. [CrossRef]
10. Oteros, J.; García-Mozo, H.; Vázquez, L.; Mestre, A.; Domínguez-Vilches, E.; Galán, C. Modelling olive phenological response to weather and topography. *Agric. Ecosyst. Environ.* **2013**, *179*, 62–68. [CrossRef]
11. Herz, A.; Hassan, S.A.; Hegazi, E.; Nasr, F.N.; Youssef, A.A.; Khafagi, W.E.; Agamy, E.; Ksantini, M.; Jardak, T.; Mazomenos, B.E.; et al. Towards sustainable control of Lepidopterous pests in olive cultivation. *Gesunde Pflanz.* **2005**, *57*, 117–128. [CrossRef]
12. Haniotakis, G.E. Olive pest control: Present status and prospects. *IOBC/WPRS Bull.* **2005**, *28*, 1.
13. Hilal, A.; Ouguas, Y. Integrated control of olive pests in Morocco. *IOBC/WPRS Bull.* **2005**, *28*, 101.
14. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
15. Raschka, S.; Mirjalili, V. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2*; Packt Publishing Ltd.: Birmingham, UK, 2019.
16. Perez, L.; Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint* **2017**, arXiv:1712.04621. Available online: https://arxiv.org/abs/1712.04621 (accessed on 11 October 2019).
17. Meier, U.; Bleiholder, H.; Buhr, L.; Feller, C.; Hack, H.; Heß, M.; Lancashire, P.D.; Schnock, U.; Stauß, R.; Van Den Boom, T.; et al. The BBCH system to coding the phenological growth stages of plants–history and publications. *J. Kult.* **2009**, *61*, 41–52.
18. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint* **2014**, arXiv:1409.1556. Available online: https://arxiv.org/abs/1409.1556 (accessed on 13 October 2019).
19. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 4278–4284.
20. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258. [CrossRef]

21. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
22. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
23. Chollet, F. *Deep Learning Mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*; MITP-Verlags GmbH & Co. KG.: Bonn, Germany, 2018.
24. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint* **2016**, arXiv:1603.04467. Available online: https://arxiv.org/abs/1603.04467 (accessed on 20 October 2019).
25. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the ICML, Atlanta, GA, USA, 16–31 June 2013; Volume 30, p. 3.
26. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
27. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint* **2015**, arXiv:1502.03167. Available online: https://arxiv.org/abs/1502.03167 (accessed on 10 October 2019).
28. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
29. Asghari, M.H.; Jalali, B. Edge detection in digital images using dispersive phase stretch transform. *Int. J. Biomed. Imaging* **2015**. [CrossRef]
30. Jalali, B.; Suthar, M.; Asghari, M.; Mahjoubfar, A. Time Stretch Inspired Computational Imaging. *arXiv preprint* **2017**, arXiv:1706.07841. Available online: https://arxiv.org/abs/1706.07841 (accessed on 20 September 2019).
31. Ang, R.B.Q.; Nisar, H.; Khan, M.B.; Tsai, C.Y. Image segmentation of activated sludge phase contrast images using phase stretch transform. *Microscopy* **2019**, *68*, 144–158. [CrossRef] [PubMed]
32. Masters, D.; Luschi, C. Revisiting small batch training for deep neural networks. *arXiv preprint* **2018**, arXiv:1804.07612. Available online: https://arxiv.org/abs/1804.07612 (accessed on 20 October 2019).
33. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint* **2014**, arXiv:1412.6980. Available online: https://arxiv.org/abs/1412.6980 (accessed on 10 October 2019).
34. Dozat, T. Incorporating Nesterov Momentum into Adam. In Proceedings of the 4th International Conference on Learning Representations, Workshop Track, San Juan, Puerto Rico, 2–4 May 2016.
35. Rojo, J.; Pérez-Badia, R. Models for forecasting the flowering of Cornicabra olive groves. *Int. J. Biometeorol.* **2015**, *59*, 1547–1556. [CrossRef]
36. Mancuso, S.; Pasquali, G.; Fiorino, P. Phenology modelling and forecasting in olive (Olea europaea L.) using artificial neural networks. *Adv. Hortic. Sci.* **2002**, *16*, 155–164.
37. Barzman, M.; Bàrberi, P.; Birch, A.N.E.; Boonekamp, P.; Dachbrodt-Saaydeh, S.; Graf, B.; Hommel, B.; Jensen, J.E.; Kiss, J.; Kudsk, P.; et al. Eight principles of integrated pest management. *Agron. Sustain. Dev.* **2015**, *35*, 1199–1215. [CrossRef]