ARCHITECTURE AND BASIC PRINCIPLES OF THE MULTIFUNCTIONAL PLATFORM FOR PLANT DISEASE DETECTION

Pavel Goncharov [2], Gennady Ososkov [1], Andrey Nechaevskiy [1], Alexander Uzhinskiy [1]

[1] *Joint Institute for Nuclear Research*

[2] *Sukhoi State Technical University of Gomel, Belarus.*

E-mail: auzhinskiy@jinr.ru

Increasing number of smartphones and advances in deep learning field opens new opportunities in the crop disease detection. The aim of our research is to develop a multifunctional platform that uses modern organization and deep learning technologies to provide a new level of service to farmer's community. Like the product, we are going to develop a mobile application allowing users to send photos and text description of sick plants and get the cause of the illness and its treatment. We have collected a special database of grape leaves consisting of five set of images. We have reached 99% accuracy in the disease detection with a deep siamese convolutional network. We have developed a web-portal with a basic detection functionality and provided an opportunity to download our self-collected image database. Concepts, an architecture of the plant diseases detection platform and a description of the used deep learning models and techniques are presented.

Keywords: machine learning, statistical models, siamese networks, plant disease detection, transfer learning

1. Introduction

Quality of available data about an impact of plant diseases is variable, patchy and often missing, particularly for smallholders, who produce the majority of the world's food. There is an opinion that crop losses by diseases are between 20 and 40% [1]. It is clear that plant diseases are a serious threat to the well-being of rural families, to the economy and governments, and to food security worldwide. The aim of our research is to facilitate the detection and preventing diseases of agricultural plants by both deep learning and programming services. We have created the plant disease detection platform (PDDP) from the scratch and share our experience. Below we describe the used deep learning models and techniques, also we give the link to our open image database.

There are many kinds of research in which deep learning was used to identify plant diseases. Some of them report about great detection level, more than 90%. However, there is a lack of a real application or open databases to reproduce the experiments. Probably, the most famous mobile application for plants disease detection is Plantix [2]. The application is developed by PEAT, a German-based AgTech startup. Currently, Plantix can detect more than 300 diseases. Plantix image database is closed and we cannot find any information about technologies used in it for disease detection. We made a small experiment processing different types of images from our self-collected database with the help of Plantix. It allows us to conclude that Plantix identification of the plant type is rather good: 64 of 70 images were recognized as grapes. At the same time, the disease detection ability is rather limited. Only a few images were detected correctly. Less than 10% of images had a right disease at the top of suggestions. Perhaps, our dataset does not match some requirements of the Plantix application. We used original images from the Internet and preprocessed ones in which problems were obvious, but the result was quite similar.

We have to reach the same functionality as Plantix but with better accuracy in the disease detection, thus we had to find an image database and a good statistical model.

2. First experiments

We considered different models that were used in the related works to understand what is the best option. In [3] authors reached high accuracy in detection up to 99.7% on a held-out test set. However, results on the real-life images were quite unsatisfactory, about 30% only. The authors have used PlantVillage [4] well known public database of 86,147 images of diseased and healthy plant leaves. It was only one public database, and we believed that we could improve their results. Eventually, we have prepared the small test set of 256x256 pixel images consisting of healthy leaves, and images with Esca, Chlorosis and Black Rot diseases and started to work.

We applied the transfer-learning approach to train a deep classifier on the Plant Village images and then evaluated it on a test subset of images. To find the most appropriate pretrained network for the transfer learning we compared four models, which weights were formed to solve the ILSVRC 2015 (ImageNet Large Scale Visual Recognition Challenge). They were VGG19 [5], InceptionV3 [6], ResNet50 [7] and Xception [8].

The comparative scheme of each classifier is to compose all layers of the trained networks except the final classification layer and to add the global average pooling operation on the top of each base network to reduce the spatial dimensions of a three-dimensional output tensor. Further, we appended a densely connected layer with 256 rectified neurons with dropout having rate of 0.5. At the end of such network, the softmax classification layer was utilized.

We froze all layers in the base networks and trained only last three layers using the stochastic gradient descent (SGD) with the learning rate equals to 5e-3, momentum 0.9 and the weight decay with the value of 5e-4 for 50 epochs. The best result of the classification accuracy with the value of 99.4% on a test subset of the PlantVillage dataset was obtained using ResNet50 architecture. We applied this model to deduce the classification efficiency on a test subset of images collected from the Internet. The obtained results were very poor – 48% accuracy.

Supposing that pretrained on the ImageNet dataset network does not extract meaning features from leaves images, we decided to unfreeze more layers. We unfroze all layers except first 140 in the base network and trained the remaining 39 defrozen layers with the help of Adam optimizer [9] with

the learning rate equals to 5e-5 and the weight decay with the value of 1e-6 for 30 epochs. Next, we proposed to apply a strong data augmentation by adding random transformations such as shifts, rotations, zooming etc., because the classification network overfits when we train more than 30 epochs. Also, we supposed that only a central part of the leaf is required to recognize disease and we tried to expand our dataset by using only parts of initial images. Some other experiments were done with background modification and other optimization, but they improved the accuracy only a little.

We believe that the problem lies in the nature of the used images. PlantVillage photos were collected and processed under special controlled conditions, so they are rather synthetic and differ from real-life images, see fig 1.

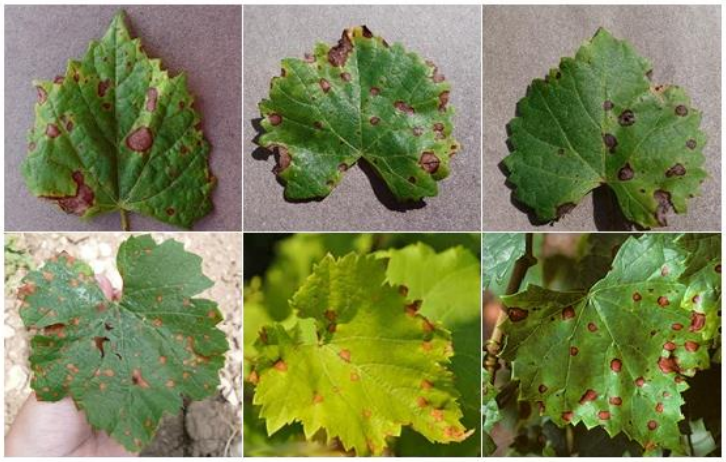

Figure 1. Top three photos are from the PlantVillage database. Bottom photos are real-life photos of sick leaves

This proves that if we want a good result, we need a real-life database. We collected our own database of the grape leaves images from open source, then reduced their size and extracted only meaningful parts. At that time, we have a set of 256x256 pixel images consisting of 130 healthy leaves, and 30 – 70 images with Esca, Chlorosis and Black Rot diseases. The number of images is very small so we have to try some new approaches.

3. Siamese networks

How could we get good features from a very small amount of data? We addressed this problem to the so-called one-shot approach offering a solution by siamese neural networks [10-13]. The siamese network consists of twin networks joined by the similarity layer with the energy function at the top. Weights of twins are tied, thus the result is invariant and in addition guarantees that very similar images cannot be in very different locations in a feature space, because each network computes the same function. The similarity layer determines some distance metric between the so-called embeddings, i.e. the high-level feature representations of input pairs of images. Training on the pairs means that there are quadratically many possible samples to train the model on, thus making it hard to overfit. We can easily compute the number of possible pairs using the combinatorics formula of k-combinations. Thereby, for the smallest class with 31 images (Black rot) we have 1860 pairs.

Our classification model is the siamese network with convolutional twins, which tie weights between themselves. Each of twins processes one image from input pair of samples to extract a vector of high-level features. Then, a pair of embeddings is passed through the lambda layer, actually computing the simple element-wise L1-distance (Figure 2). We present the connection a single sigmoid neuron to the distance layer thus it becomes possible to train the model with binary cross-entropy loss.

We use exclusively rectified linear (ReLU) units in the twins except for the last densely-connected layer with sigmoid activations. The initial number of convolutional filters is 32 and doubling each layer. After the last pooling layer, we added a flatten operation to squeeze the convolutional features into a vector followed by a densely-connected layer with 1024 sigmoid neurons.

We also experimented with adding the L2 regularization to each of the convolutional layers but have not received a visible effect. In addition, we have been trying to vary the size of the embedding layer from 256 to 4096. The best architecture we created is presented in figure 2.

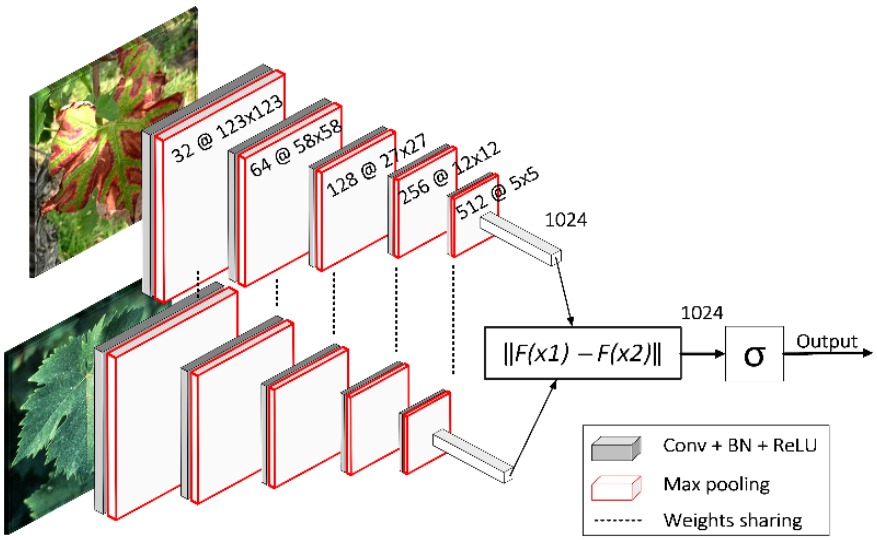

Figure 2. Our best siamese convolutional architecture. «Conv» means the convolutional operation, «BN» is a batch normalization, «32 @ 123x123» – 32 feature maps with the particular size

We used the K-nearest neighbors (KNN) algorithm to solve the classification task in a test phase. We set the K parameter to 1 nearest neighbor, so it is equivalent to one-shot learning task, except one moment – we utilize all training data as a support set instead of the random picking from the dataset. For the distance metric, we have used the absolute distance in the lambda layer of the siamese network, we preferred to manage the manhattan distance.

We apply the data augmentation by adding rotations in range of 75 degrees, random shifts in all dimensions including a shift within a channel in the range of 0.2. In addition, we use little zooming, vertical and horizontal flips. We have trained the siamese network with the proposed augmentation for 100 batches size 32 per one training epoch for 35 epochs. As the method of optimization, we used Adam with the learning rate value of 0.0001. For the loss function, we utilized the binary cross-entropy loss.

After training the model, we left only the encoder network represented as «shoulder» of the one-shot model, or so-called twin. We further used this part of the network as a feature extractor. After that, we took the training subset with five classes of images: Healthy, Esca, Black rot, Chlorosis, and Mildew. We split data on train and test with the ratio near 75:25. Then we passed these sets of images through our feature extractor to obtain data embeddings. The training subset of images is then utilized as training data for the KNN. For verification, we used the remaining test set. The classification accuracy was up to 94%. Besides, we tried to mix the real-life data with the images from the PlantVillage database within train and test subsets. The Siamese networks allow generalizing input data to the latent high-dimensional embedding space, even for unseen classes. The obtained accuracy with the value of 92% (our best result) proves this consideration.

We used the prepared embeddings to train the T-SNE method [14], which is the common technique to visualize high-dimensional data. We extracted two components to plot them in 2D space (Figure 3). One can see that there are five separate clusters – one per each class. We have signed each cluster with the name of a particular class.

There are a few points, which wrongly got into the different set, we investigated such cases and slightly modified the images. For example, one healthy leave on the white wall was at the mildew cluster, so we removed the background influence of the image, and it moved to the healthy cluster. After that, we trained model again and got 99% accuracy.

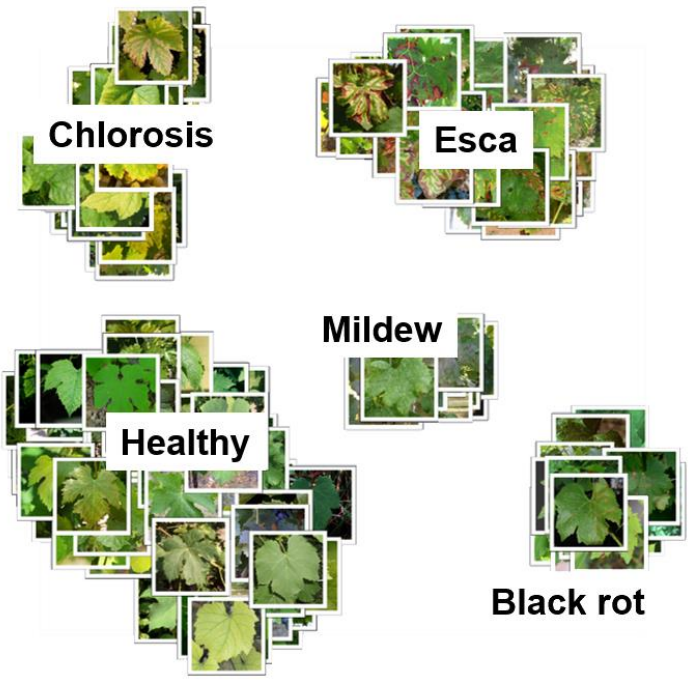

Figure 3. T-SNE visualization of the high-level features extracted by the siamese twin

3. Architecture and principles of the platform

After finishing the creation of our base network, we moved to other parts of the platform. We agreed on the architecture presented in Figure 4.

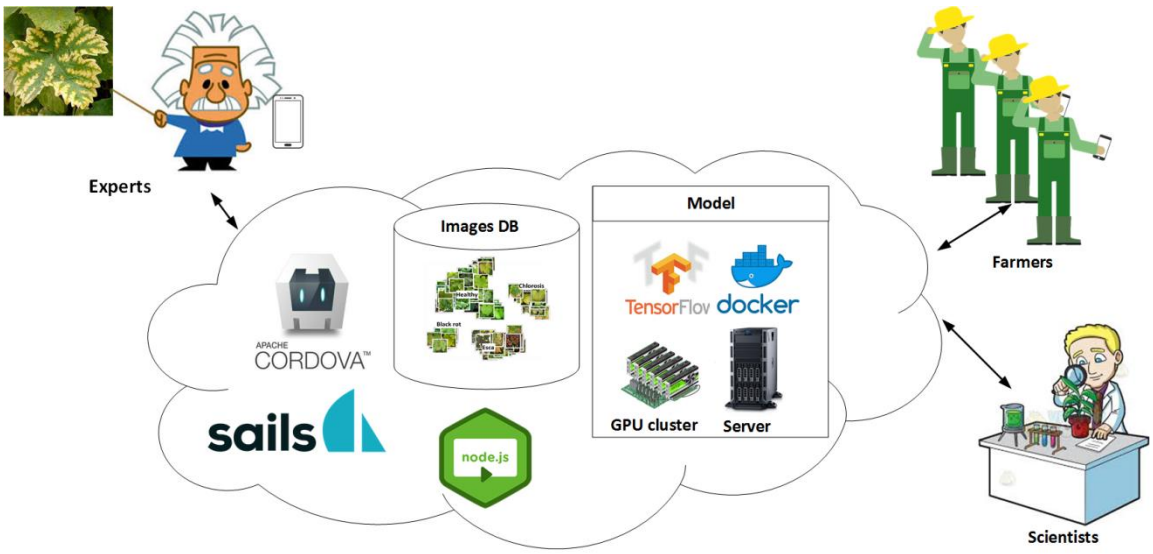

Figure 4. Architecture of the PDDP

The PDDP consists of a set of interconnected services and tools developed, deployed and hosted with the help of the JINR cloud infrastructure. Our web-portal (pdd.jinr.ru) was developed with the Node.js (Sail.js). It provides not only a web-interface but also the API for third-party services. We have the TensorFlow model in the Docker realized as a service. The model can work at the virtual server, or at a GPU cluster. Right now, we are storing images directly on the local drive, but if their number increases dramatically, we will use cloud storage like disk.jinr.ru. We will use the Apache Cordova to create the Mobile App for Android, IOS and Windows platforms.

We also determined basic roles and use-cases for PDDP. Users can: send photos and a text description of sick plants through the web interface, or mobile application and get the cause of the

illness; browse through disease's description and galleries of ill plants; verify that the requested disease was recognized right and the treatment helps. Experts can: browse users' requests and verify the recognition; request an addition of their image, or image from the user complain to DB; request an change in the diseases description; request a retraining of the model with new images. Researchers can: work with images database through the web-interface, or API; download all or only a part of the base; obtain the API-key to submit recognition tasks to the platform. Supervisors can: add new images to the database; initiate a retraining of the model; get different statistical metrics about portal users.

Currently, we have a website with general information – pdd.jinr.ru; open image database; model running in docker container at the virtual server. One has the ability to submit disease detection jobs at the website and get results, see Figure 5

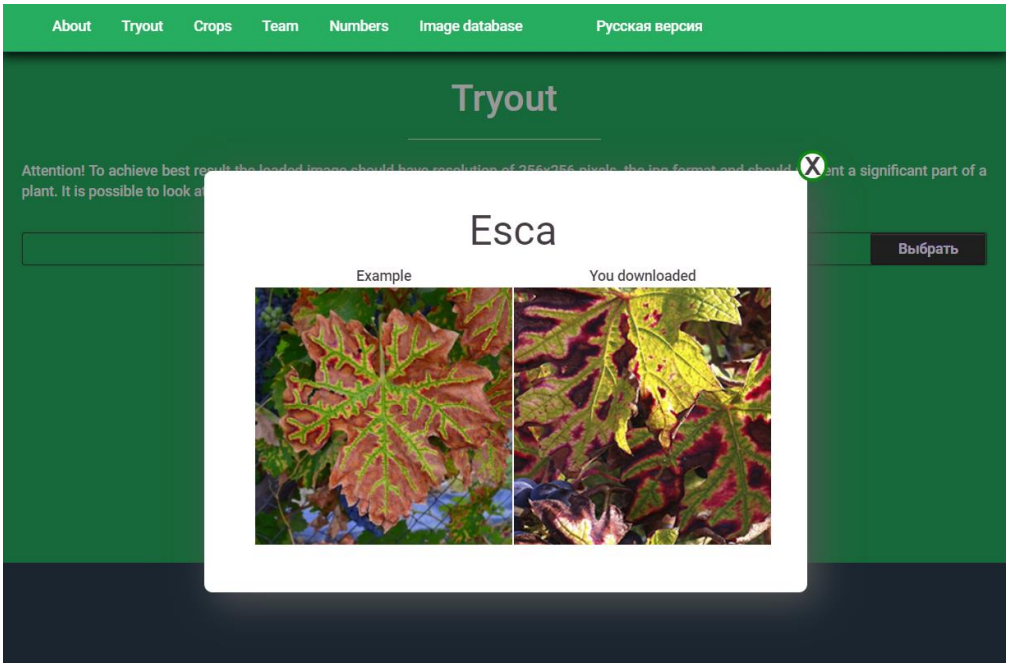

Figure 5. Results of the grape diseases detection at pdd.jinr.ru

4. Conclusion and plans

It is not enough to have a lot of images to recognize diseases rightly. Quality of the images database is extremely important for results of detections. The Siamese neural networks are very perspective research field for the plant disease detection projects. We have reached 99% accuracy in the detection of some grape diseases. We will keep on developing our web-portal, and we are going to present a draft mobile application in the second half of 2019.

Acknowledgment

The reported study was funded by RFBR according to the research project № 18-07-00829

References

[1] Serge Savary & Andrea Ficke & Jean-Noël Aubertot & Clayton Hollier, Crop losses due to diseases and their implications for global food production losses and food security// Springer Food Security, Vol 4, 2012

[2] Plantix project home page [Electronic resource]: https://plantix.net. (Accessed 1.10.2018)

[3] Mohanty S.P., Hughes D.P., Salathé M.: Using Deep Learning for Image-Based Plant Disease Detection, Frontiers in Plant Science 7. Article: 1419 (2016)

[4] PlantVillage project home page [Electronic resource]: https://plantvillage.psu.edu/. (Accessed 1.10.2018)

[5] Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition //arXiv preprint arXiv:1409.1556. – 2014.

[6] Szegedy C. et al. Rethinking the inception architecture for computer vision //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2016. – p. 2818-2826.

[7] He K. et al. Deep residual learning for image recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – p. 770-778.

[8] Chollet F. Xception: Deep learning with depthwise separable convolutions //arXiv preprint. – 2016.

[9] Kingma D.P., Ba J. Adam: A method for stochastic optimization //arXiv preprint arXiv:1412.6980. – 2014.

[10] Koch G., Zemel R., Salakhutdinov R. Siamese neural networks for one-shot image recognition //ICML Deep Learning Workshop. – 2015. – T. 2.

[11] Taigman Y. et al. Deepface: Closing the gap to human-level performance in face verification //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2014. – p. 1701-1708.

[12] Hadsell R., Chopra S., LeCun Y. Dimensionality reduction by learning an invariant mapping //Computer vision and pattern recognition, 2006 IEEE computer society conference on. – IEEE, 2006. – T. 2. – p. 1735-1742.

[13] Appalaraju S., Chaoji V. Image similarity using Deep CNN and Curriculum Learning //arXiv preprint arXiv:1709.08761. – 2017.

[14] Maaten L., Hinton G. Visualizing data using t-SNE //Journal of machine learning research. – 2008. – T. 9. – №. Nov. – C. 2579-2605.