International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 14 (2022) pp. 092-099
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Submitted: 20 September, 2021; Accepted: 6 Februay, 2022; Publish: April 21, 2022

FGVC8 based foliar diseases identification with use of multi-label classifiers

Mateusz Chiliński[1], Piotr Goralewski[2], Tomasz Lehmann[3], Adam Nowacki[3], Justyna Stypulkowska[3]

[1]Faculty of Mathematics and Information Science, Warsaw University of Technology,
Koszykowa 75, 00-662, Warsaw, Poland
m.chilinski@mini.pw.edu.pl

[2]Faculty of Electrical Engineering, Warsaw University of Technology,
Pl. Politechniki 1, 00-661, Warsaw, Poland

[3]Faculty of Electronics and Information Technology, Warsaw University of Technology,
Nowowiejska 15/19, 00-661, Warsaw, Poland

All authors have contributed equally.

Abstract: **Image recognition is achieving unprecedented success in many areas, while high requirements for the detection accuracy of objects and the characteristics of objects present in images remain difficult challenges. On the other hand, the continued popularity of Computer Vision and the constant improvement of the quality of image recognition algorithms are the best way to solve these problems. This approach became the impetus for writing this article.**
The article examines the effectiveness of using *FGVC* (Fine-Grained Visual Categorization) algorithms to obtain the best solution to the problem of recognizing apple leaf diseases. The problem is very complex, because the leaves with symptoms of diseases are often infected with many diseases at the same time, which makes the whole process difficult.
After analyzing the possible approaches to solving the problem, it was decided to examine four methods: use the most primitive AI (Artificial Intelligence) approach, use classic methods of image classification, check deep neural network based on multi-label classification and use deep neural transfer learning multi-label classification with search cutoff optimization. The conducted research using a proven and representative set of training and test data made it possible to compare all the above-mentioned methods and to draw conclusions about the accuracy of using each of the methods.

Keywords: computer vision, image recognision, machine learning, apple leaf diseases identification, artificial intelligence

I. Introduction

Foliar diseases pose a great threat to the productivity and health of apple orchards, which are one of the most popular fruit crops in the world. Usually, the detection of apple diseases is done manually by observation, which is very time-consuming. While the use of computer vision can be helpful, it has limitations that must be overcome. Visual differences in the prevalence of the same disease and new varieties are additional challenges. The main differences are related to the color of the leaves, their morphology, the age of the tissues and the lighting when collecting the images. The main challenge is to develop machine learning models to classify a leaf image from a test kit to a given disease, and to detect a specific disease despite the presence of symptoms of multiple diseases on a given leaf [1].
In more technical aspects of the presented research, it is necessary to explain what *FGVC* is. These are *Fine Grain Visual Classification* algorithms that classify classes of objects that are difficult to distinguish, such as plant species (including their diseases) or animals, and identifying brands of cars or mechanical parts. *FGVC* datasets deviate from conventional image classification, because they typically require expert knowledge, not crowdsourcing, to collect annotations. *FGVC* datasets contain images with much higher visual similarity, than in *LSVC (Large Scale Visual Classification)* [2]. Additionally, *FGVC* datasets have intrinsic inter-class visual differences, in addition to the differences in lighting and point of view that are present in *LSVC*. Furthermore, *FGVC* datasets often have long tails in data distribution because the difficulty of getting examples of different classes can vary. A combination of small, highly heterogeneous and non-heterogeneous data sets with internal differences between classes make *FGVC* a huge challenge even for advanced deep learning algorithms [3].
The combination of small, highly heterogeneous and non-heterogeneous datasets with internal class differences makes *FGVC* a huge challenge even for advanced deep learning algorithms.
During our research, we proposed four different approaches to the implementation of the problem posed. The first is based on the use of primitive AI approaches such as *Support Vector Machines (SVM)* and *Stochastic Gradient Descent (SGD)*. Another approach we have prepared is a us-

age of some classic computer vision methods of describing the image, and then learning primitive AI models with found data. The third approach is deep neural network based multi-label classification. As the last example, we used deep neural transfer learning multi-label classification with random search cutoff optimization.

II. Related works

Early studies of *Fine-Grained Visual Classification (FGVC)* were investigating training methods with limited tagged data and traditional image characteristics. *Yao et al.* [4] used the highly discriminating image fragments with randomization techniques to avoid overfitting. They used template matching to avoid a huge number of annotations.
Comparing *FGVC* datasets with *Large Scale Visual Grading (LSVC)* datasets showed that *FGVC* datasets are much smaller and more noisy than *LSVC* datasets [3].
Recently, improved target localization in training images has been shown to be useful in *FGVC* [5]. *Zhang et al.* [6] use partially based *CNN* regions [7] for more precise localization. Space transformer networks [8] prove that learning the content-based affine transformation layer improves *FGVC* performance. Position normalized CNN has also been shown to be effective in *FGVC* [9]. Model building and amplification also improved performance in *FGVC* [10]. *Lin et al.* [2] introduced *Bilinear Pooling*, which pairs local trait sets and improves classification performance. *Gao et al.* [5] extended *Bilinear Pooling* by using a compact bilinear representation, while *Cui et al.* [11] by using a generic kernel-based pool structure that captures higher-order trait interactions.
An important point to mention is pair learning. *Chopra et al.* [12] introduced a Siamese neural network that is used to recognize handwriting. *Parikh* and *Grauman* [13] developed a pairwise ranking scheme to accomplish relative attribute learning. Later, pairwise neural network models were often used to model features.
Another issue that should be brought closer is Learning from Label Confusion. This method aims to improve classification efficiency by confusing output labels. Previous work in this area includes methods using label noise [7] and data noise during training. *Krause et al.* [14] used noisy training data for *FGVC*. *Neelacantan et al.* [15] added noise to the gradient during training to improve the efficiency of the generalization process in deep networks. *Szegedy et al.* [16] introduced label-smoothing regularization to train deep inception models.
In reference to above research, we decided to combine the concepts of pair learning and label confusion in order to solve the problems of overfitting and applying selected artifacts when training neural networks in the realization of *FGVC* examples [3].

III. Dataset

During our research, we worked on a dataset made available on the *Kaggle* competition website. This dataset contains photos of apple leaves infected with number of common diseases, including several on the surface of the same leaf. Each photo is labeled as healthy or with one or more of diseases. A complete dataset consists of two csv files and two foleders with images files:

- train.csv (the training set metadata) containing two columns: image (with the image IDs) and labels (with the target classes, a space delimited list of all diseases present on the surface of leaf shown in the corresponding photo);
- sample-submission.csv (a sample submission file in the correct format) also containing two columns with the same names as in the previous file: image and labels;
- train-images - folder with the training images, totaling 18.6k images;
- test-images - folder with the test images, containing three photos.

The common apple foliar diseases which are presented in the dataset are: frogeye leaf spot, scab, rust and powdery mildew. First three are quite similar, but have some slight differences. Frogeye [17] tends to create many small spots, light brown to grey in the middle, with dark brown contour around the spot. Apple scab [18] creates less circular brown spots, which look like mist - denser in some parts and sparse in others. Apple scab spots also tend to group, creating one big, brown area on a leaf. Rust [19] also creates spots on leaves, but this time the spots are yellowish or orange to light brown, in comparison to earlier two. Finally, powdery mildew [20] is different - it covers the whole leaf (or its large part) with white or light grey powder. The powder sometimes forms small white veins. It looks like a leaf wrapped in a spider's web.
As the given train-test split, made by authors of dataset, is inadequate to the problem, we splited the dataset into three parts: train, test and validation subsets, with proportions 60:20:20. To ensure consistency, all applied approaches were trained, tested, and validated on the same three subsets.

IV. Proposed methods

This section presents four approaches to solving the problem analyzed in this article. The following subsections provide a description of independent methods.

A. *Primitive AI approach*

At the very beginning of the study, we decided to check how the most classical and primitive approach will perform. The images were loaded using the *Python Imaging Library* [21], and resized to 200x200 *RGB* picture, totalling to a vector of 120,000 numbers each. No additional filtering was performed in this approach, and we have decided to test two very common approaches - *Support Vector Machines (SVM)* and *Stochastic Gradient Descent (SGD)*. The similar approach will be presented in the next subsection, however, with additional processing of the data to improve performance.
In case of the unfiltered data, the large input has become problematic as the classical *SVC* has $o(n^2)$ computational complexity [22], an optimisation had to be introduced. We wanted to test all the kernels available in the *scikit-learn* package, but the calculation time of such a huge input (more

than 15,000 images, 200x200x3 each) because of the aforementioned property of *SVC* was impossible. That is why we have used *LinearSVC* - optimised version of *SVC* for the linear kernel case.
The lack of other kernels made us search for another classic approach - that is why we decided to test the *SGD* approach, which is also considered to be a classic one in the artificial intelligence field, however both approaches are not suitable for feature extraction, so we expected to see them being outperformed by the deep learning approach.
The approach creates independent models for each of the features, and performs one-vs-all prediction. Each of the feature is predicted independently, and then the results are put together.
Next approach is similar to this one, however instead of learning the AI algorithms with raw photos, we tried to use knowledge about diseases to extract some data from the photos.

B. Classic methods of image classification

In this approach we wanted to check how well classic methods of image classification would behave in such a complex task. In comparison to the previous approach, here we try to use outer knowledge about the issue, to extract data from images, and then use this data to learn AI models. We did not expect this approach to achieve very good results, but we wanted to show how far the technology went through recent years, and how complex the problem is.
The idea was to analyze apple foliar diseases and exploit found information to create descriptors able to describe them. In our task we consider four apple foliar diseases, which were described in III.
Based on knowledge about the diseases, we can expect that photos of sick leaves will have more brownish spots, more white color or overall more contours than photos of healthy leaves. To use this information we decided to use some descriptors of the photos. First and most intuitive: histogram of colors (histogram of hue values in the photo converted to *HSV*). We expect to see regular differences in color distributions, as all of the diseases affects surface of leaves changing their color from green to brownish or greyish. To find whether there are spots on a leaf, we detect brownish blobs in the picture. This is useful information in case of frogeye, scab and rust. Finally, we check the number of edges in the image - powdery mildew tends to create small veins which might be tried to be detected. Also any disease can destroy natural "leaf veins".
Finally, we use all this data to train some well known classifiers: *Gaussian Naive Bayes*, *Multinomial Naive Bayes*, *Random Forest* and *Support Vector Machine*. As all those algorithms are not well adapted to multi-class classification, we used the *One Versus Rest* approach. In other words, we divide the problem of multi-class classification into some binary classifications. Firstly the algorithm tries to classify is the leaf diseased with scab or anything else, then frogeye or anything else, then rust or anything else etc. Finally classes which get the most confidence are taken.
Overall this approach achieved poor results. *Random Forest* turned out to be the best classifier out of the four, but still its results are below level of satisfaction. After analysis of results, we think that there are two main reasons causing such bad classification quality. Firstly, classic classifiers were not designed for multi-class classification, and they are simply too weak for such complex task. Secondly and probably more importantly, the diseases are very similar. It is hard to distinguish between frogeye, rust and scab, sometimes even for humans. Such complex task demands usage of more advanced technologies, such as deep neural networks.

C. Deep neural network based multi-label classification

In this section, we describe our method and experimental results for multi-class estimation with deep neural network architecture.
The proposed network consists of two parts, i.e., a dense feature extractor (considered as a main backbone of approach), and fully connected perceptron built of four layers. The final output is presented as a binary vector of size 9, where the first six values describes health state of analyzed foliar (the first element responds to class healthy" and the others represent every possible disease occurred in the dataset: powdery mildew, rust, scab, complex and frog eye leaf spot). Three neurons on the end of the mentioned vector are responsible for indicating how many classes belong to a given leaf. The architecture is shown in Tab. 1.

Layer	Output	Function
Backbone	9	SeNet154 last linear layer
P1	1000	First perceptron layer
Leaky ReLu	1000	Activation function
P2	1000	Second perceptron layer
Leaky ReLu	1000	Activation function
P3	9	Third perceptron layer
Sigmoid	9	Final activation function

Table 1: **Network architecture.** The first mentioned layer is the *SeNet154* last linear layer. We follow *P1* and *P2* fully connected layers by a leaky *ReLU* activation function [23] with parameter $\alpha = 0.1$. At the end we use sigmoid function. Note that in this table we called dimension of the single returned tensor (number of neurons in the network last layer) as an *output*.

The input RGB image was encoded into a feature vector using the *SeNet154* [24] network which was firstly pretrained on *ImageNet* dataset [25]. For training our network we selected the *Mean Square Error (MSE)* loss function:

$$MSE = \frac{1}{n}\sum_{n}(x_n - y_n)^2$$

$n -$ number of vectors

$x_n -$ vector ground truth

$y_n -$ vector prediction

Figure. 1: MSE Algorithm

ADAM [26] was chosen as the optimizer..
Our model was trained on a subset of 11178 images. The input images were scaled to size 224x224. Because of an imbalanced dataset during training we used a random undersampling method [27]. It means that in every step of

the learning process we randomly selected examples from the majority classes and deleted them. As a result, in every epoch the amount of elements in every class was the same. The method was used to increase sensitivity in minority classes prediction. We implemented our architecture using *PyTorch* [28] and trained on *NVIDIA TITAN V100* with 32 *GB* memory. Batch size was fixed at 32. Number of training iterations was fixed at 200 epochs. In Fig. 2, we can see the training loss decreasing during the first 30 epochs and then starting to normalize.

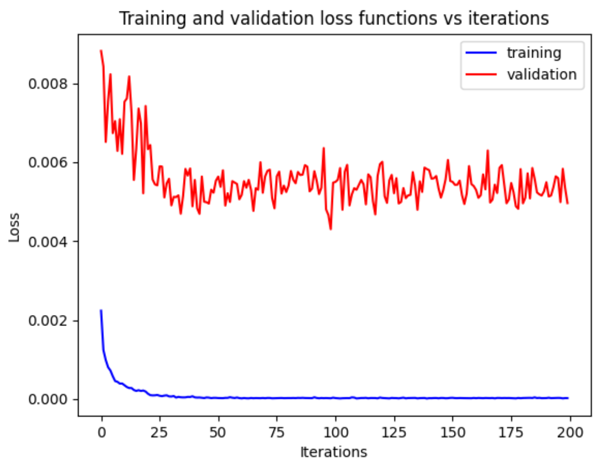

Figure. 2: **Comparing of training and validation loss.** The horizontal axis represents the number of training iterations (in epochs). The vertical axis represents the loss of the validation and training set at each epoch during a single training process.

After every training epoch we validated the newest model on the validation subdataset which contained 3727 elements. During the validation we also used an undersampling method but this time we focused on multi-labeling instead of diseases. In every epoch we made the validation dataset to contain the same number of one-label, two-label and three-label elements. In this way we wanted to increase sensitivity on the last three neurons of the returned output. Because of that change the loss function of the validation shown in Fig. 2 is much more uneven then the loss function of the training. In this way we optimized model in disease and multi-label classification.

Validation process was supposed to determine optimal values of thresholds above which an object belongs to two-label or three-label category. For this purpose we examined the average percent of validation samples classified to correct multi-label class by changing values of mentioned thresholds in the range between 0.01 and 0.2 with step 0.02. The given metric was optimal when both values were equal to 0.1. That means if the value of 8th neuron in the single prediction is above that – the analyzed plant has at least two diseases. In the situation when the value of the last neuron is bigger than 0.1 – the object is classified into three classes. The exception is the situation when the first class is predicted as the healthy" one. Then thresholds are not considered at all.

The explained model have also been tested with a "backbone" based on *Resnet50* [29] and *Xception* [30] architectures but both results were slightly worse than the firsts one.

D. Deep neural transfer learning multi-label classification with random search cutoff optimization

In this approach we take advantage of Deep Neural Networks for multilabel image classification because used dataset contains photos assigned to one class or many classes at the same time. At the beginning, the data was divided into a training, validation and test set in the proportion of 3:1:1 respectively as mentioned in III. Initial inbalanced proportion of classes and labels distribution between the separated data subsets was maintained. For all implementation stages the random seed was set to 42 to ensure pipeline repeatability and reproducibility.

Code implementation, model trainings and validations, optimizations, and all other developmental stages of presented approach were completed using a private computing *PC* with 32 *GB RAM*, 9th generation *CPU i9*, and *GPU RTX 2070 8 GB*. Implemented pipeline in *Python* programing language required the use of the following libraries: pandas [31], numpy [32], random, shutil, os, tensorflow-gpu [33], keras [34] and glob.

To achieve the best results in multilabel classification we used a transfer learning approach to utilize certain Deep Learning architectures. This is a common practice in image processing. In most Computer Vision applications pretrained architectures allow to achieve better results than the vast majority of models implemented from scratch. To verify the effectiveness of individual models and obtain the best possible outcome, we used architectures such as: *DenseNet121* [35], *DenseNet169* [35], *DenseNet201* [35], *EfficientNetB0* [36], *EfficientNetB1* [36], *EfficientNetB2* [36], *Inception ResNet* [37], *Inception V3* [38], *MobileNet V2* [39], *NasNet-Large* [40], *NasNetMobile* [40], *ResNet50*, *ResNet101* [29], *ResNet152* [29], *VGG16* [41], *VGG19* [41] and *Xception*. All the pretrained model's architectures with optimized weights on the *ImageNet* dataset were imported from the *Keras* model repository [42]. The top layer from source architecture was replaced by the *Global Average Pooling 2D* and dense layers. In this approach, we performed a multi-label classification, in which each photo is classified into 6 labels (scab, rust, frog eye leaf spot, powdery mildew, healthy, complex) and for each of them the probability of belonging is returned independently. This solution required the use of a sigmoidal activation function in last dense classification layer – produced probability for each class is in the range of 0-1. Setting a specific cutoff combination we adjust the model which ultimately classifies an image into one class, many classes at the same time or none of them, rendering an inconclusive result.

The model training and validation process was divided into two main stages. Implementation required the use of the *ADAM* optimizer as optimization algorithm, binary crossentropy as the loss function, and an accuracy metric as the effectiveness assessment measure. In the first stage, the weights of the pre-trained model were frozen and only the parameters of the output dense layer were optimized. In the second stage, the coefficients of the base model were unfrozen and the weights of the entire model were optimized without any restrictions.

The data augmentation process was used in both stages and

was responsible for performing random modifications and transformations (rescale, horizontal flip, vertical flip, rotation, shear and zoom) on input tensors, which allows for counteracting overtraining of the model and increases its effectiveness in generalizing input information. When looking from the perspective of the final result, the second phase of training proved to be more important, hence it was scheduled for a much longer timeframe - 50 epochs, by default. To prevent model overfitting, an early stop criterion was implemented. This interrupted the training process of the model if the binary crossentropy loss value did not improve for more than 5 consecutive epochs. After each epoch, a verification process was conducted on the separated validation data. If the model has a lower loss value after an epoch, and the result has improved, the current best combination of parameters of the trained model was automatically recorded and saved as model the best checkpoint. After activating the early stopping criterion, the best model was restored for further works from a saved file containing optimized weights. At the end, best tuned parameters were evaluated on all three datasets (to ensure the comparability of the results, the training, validation and test sets are identical for all presented solutions) and predicted probabilities for all of images were saved for next project purposes.
In the proposed approach, important factors influencing the obtained final results are the cutoff values for individual classes to which the photos are classified. To maximize the values of specific metrics, the cutoffs were optimized via a random search method. For each class, values were randomly generated from a uniform distribution from 0 to 1, which act as decision thresholds - the probability above the cutoff classified a given photo to the positive class, and below, to the negative one. To find the optimal thresholds for each label in the validation set, the process was repeated 1000 times for all classes simultaneously. Values of measures such as accuracy, precision, recall and F1 were calculated and accounted for in cutoff combinations. This method found threshold values maximizing the specified metric on all classes and imposed them on test probabilities. Ultimately, the results obtained in this method on the test set are usually better than those found by using the default decision threshold values. Due to the strong imbalance of the classes, thresholds were selected to maximize the F1 measure within each label on validation dataset.

V. Comparison of experiments results

This section presents comparison of experiments results come from the four approaches to solving the problem analyzed in this article. The following subsections provide a results of independent methods.

A. Accuracy and F1 class metrics

To ensure full analysis of the results, the accuracy and F1 metrics were also calculated for all models and classified diseases. SeNet154 deep neural network based multi-label classification, with few exceptions, turned out to be the most effective for both metrics. The second most successful approach was Xception deep neural network with random search cutoff optimization.

The obtained results for images positively qualified to one-class category confirm that approaches based on deep neural networks have enormous potential.

- *SGD*

Disease	Precision	Recall	F1
healthy	0.42	0.24	0.31
powdery mildew	0.35	0.20	0.25
rust	0.24	0.08	0.12
scab	0.61	0.35	0.44
complex	0.42	0.54	0.47
frog eye leaf spot	0.25	0.73	0.37

Table 2: **Primitive AI approach - SGD.** Precision, Recall and F1 score for elements positively classified as an one-class type. *Accuracy = 36*

- *Random Forrest*

Disease	Precision	Recall	F1
healthy	0.68	0.76	0.72
powdery mildew	0.93	0.91	0.92
rust	0.75	0.84	0.79
scab	1.00	0.32	0.48
complex	0.76	0.72	0.74
frog eye leaf spot	0.77	0.60	0.67

Table 3: **Classic methods of image classification - Random Forrest.** Precision, Recall and F1 score for elements positively classified as an one-class type. *Accuracy = 76*

- *SeNet154*

Disease	Precision	Recall	F1
healthy	0.97	0.99	0.98
powdery mildew	0.98	0.97	0.98
rust	0.94	0.97	0.95
scab	0.94	0.97	0.96
complex	0.92	0.64	0.75
frog eye leaf spot	0.94	0.97	0.95

Table 4: **Deep neural network based multi-label classification - SeNet154.** Precision, Recall and F1 score for elements positively classified as an one-class type. *Accuracy = 95*

- *Xception*

Disease	Precision	Recall	F1
healthy	0.97	0.96	0.97
powdery mildew	0.94	0.97	0.95
rust	0.95	0.99	0.97
scab	0.99	0.98	0.98
complex	0.97	0.99	0.98
frog eye leaf spot	0.93	0.69	0.79

Table 5: **Deep neural transfer learning multi-label classification with random search cutoff optimization - Xception.** Precision, Recall and F1 score for elements positively classified as an one-class type. *Accuracy = 96*

ID	Disease	Accuracy	Model
1.	complex	0.930507	SeNet154
2.	complex	0.926482	Xception RS
3.	complex	0.925946	ResNet50
1.	frog eye leaf spot	0.941776	SeNet154
2.	frog eye leaf spot	0.933727	Xception RS
3.	frog eye leaf spot	0.929434	ResNet50
1.	healthy	0.988463	SeNet154
2.	healthy	0.985779	Xception RS
3.	healthy	0.983901	ResNet50
1.	powdery mildew	0.995170	SeNet154
2.	powdery mildew	0.991682	Xception RS
3.	powdery mildew	0.990877	ResNet50
1.	rust	0.982291	SeNet154
2.	rust	0.982291	ResNet50
3.	rust	0.978535	Xception RS
1.	scab	0.940971	SeNet154
2.	scab	0.939361	Xception RS
3.	scab	0.932117	InceptionResNet RS

Table 6: Top accuracy per class models.

ID	Disease	F1	Model
1.	complex	0.706638	Xception RS
2.	complex	0.689076	SeNet154
3.	complex	0.685649	ResNet50
1.	frog eye leaf spot	0.879109	SeNet154
2.	frog eye leaf spot	0.864658	Xception RS
3.	frog eye leaf spot	0.852661	ResNet50
1.	healthy	0.977091	SeNet154
2.	healthy	0.971521	Xception RS
3.	healthy	0.968254	ResNet50
1.	powdery mildew	0.964981	SeNet154
2.	powdery mildew	0.939335	Xception RS
3.	powdery mildew	0.934109	ResNet50

ID	Disease	F1	Model
1.	rust	0.925843	SeNet154
2.	rust	0.925170	ResNet50
3.	rust	0.907621	Xception RS
1.	scab	0.899818	SeNet154
2.	scab	0.892278	Xception RS
3.	scab	0.879351	InceptionResNetV2 RS

Table 7: Top F1 per class models.

B. Proposed multi-label metrics

In this section we compere only solutions based on neural networks which far surpassed traditions methods. Given the complexity of the multi-labeling classification we made the decision to use following metrics to evaluate our models on the test dataset.

1. Confusion matrix for multi-label recognition.

   - *SeNet154*

predicted as →	one-class	two-class	three-class
one-class element	97%	1%	2%
two-class element	54%	15%	30%
three-class element	51%	6%	43%

Table 8: **Deep neural network based multi-label classification - SeNet154.** Confusion matrix for multi-label elements recognition.

   - *Xception*

predicted as →	one-class	two-class	three-class
one-class element	91%	5%	1%
two-class element	45%	37%	17%
three-class element	31%	31%	37%

Table 9: **Deep neural transfer learning multi-label classification with random search cutoff optimization - Xception.** Confusion matrix for multi-label elements recognition.

2. Additional metrics to examination of multi-label classification quality might be interpreted as a special type of recall and precision where predicted element is qualified as a *true positive (TP)* only when whole vector is predicted correctly.

   - *SeNet154*

Percent of correct predictions for one-label elements	92%
Percent of correct predictions for one-label predictions	91%
Percent of correct predictions for two-label elements	14%
Percent of correct predictions for two-label predictions	40%
Percent of correct predictions for three-label elements	43%
Percent of correct predictions for three-label predictions	10%
Total percent of true predictions	87%

Table 10: **Deep neural network based multi-label classification - SeNet154.** Additional metrics.

   - *Xception*

Percent of correct predictions for one-label elements	88%
Percent of correct predictions for one-label predictions	93%
Percent of correct predictions for two-label elements	19%
Percent of correct predictions for two-label predictions	15%
Percent of correct predictions for three-label elements	37%
Percent of correct predictions for three-label predictions	15%
Total percent of true predictions	83%

Table 11: **Deep neural transfer learning multi-label classification with random search cutoff optimization - Xception.** Additional metrics.

The achieved results showed that deep neural network based methods were far better then the classical ones. The main, still not completely solved, problem was to recognize single-class and multi-class elements. According to our latest knowledge - the diseases classification should be considered as a state-of-the-art on this *Kaggle* dataset.

VI. Conclusions

After presenting and comparing the results of the experiments from the four approaches to solving the problem analyzed in this article, one can draw some important conclusions about the effectiveness of tested methods. Methods that were used during the research were: primitive AI approach - *SGD*, classical methods of image classification - *Random Forrest*, deep neural network based multi-label classification - *SeNet154* and deep neural transfer learning multi-label classification with random search cutoff optimization - *Xception*. In order to provide a complete analysis of the results, in addition to recision and recall, accuracy and F1 metrics for all models and classified diseases were also calculated. The

multi-label classification based on the *SeNet154* deep neural network, with a few exceptions, proved to be the most effective for both metrics. The second most successful approach was the *Xception* deep neural network with random search cutoff optimization.
In such complex tasks, methods based on deep neural networks have great potential and are much better than classical methods of image classification. This is confirmed by the results obtained for images positively classified as one-class category. However, the problem in both cases was still not fully resolved because recognizing single-class and multi-class elements was difficult. Additionally, the problem of distinguishing objects belonging to many classes between objects belonging to one class is extremely difficult, and the difficulty level increases, especially if the classes are similar to each other.
An important conclusion from the work done is that in the classical approaches to solving the problem presented in this article, it has been noticed that the use of descriptors (rather than a raw image) increases accuracy. Despite this, however, solutions based on neural networks are far superior to traditional methods. Given the complexity of the multi-label classifications, two metrics were used to evaluate the models used. They were Confusion matrix for multi-label recognition and special type of recall and precision where predicted element is qualified as a *true positive (TP)* only when whole vector is predicted correctly.
The achieved results showed that deep neural network based methods were far better then the classical ones. The main, still not completely solved, problem was to recognize single-class and multi-class elements. According to our latest knowledge - the diseases classification should be considered as a state-of-the-art on this *Kaggle* dataset.
An additional important conclusion related to the analysis of the works related to the subject of this article is that machine learning and computer vision evolve very quickly and progress in this field is taking place with great dynamics. The problems that are posed today are likely to be resolved in a relatively short time.

Acknowledgments

This work has been supported by National Science Centre, Poland (2019/35/O/ST6/02484).

References

[1] https://www.kaggle.com/c/plant-pathology-2021-fgvc8, 2021.

[2] RoyChowdhury A. Maji S. Lin, T.Y. Bilinear cnn models for fine-grained visual recognition., 2015.

[3] Guo P. Raskar R. Farrell R. Naik N. Dubey A., Gupta O. Pairwise confusion for fine-grained visual classification, 2018.

[4] Khosla A. Fei-Fei L. Yao, B. Combining randomization and discrimination for fine-grained image categorization., 2011.

[5] Beijbom O. Zhang-N. Darrell T. Gao, Y. Compact bilinear pooling., 2016.

[6] Donahue J. Girshick-R. Darrell T. Zhang, N. Part-based r-cnns for fine-grained category detection., 2014.

[7] Lee H. Anguelov-D. Szegedy C. Erhan D. Rabinovich A. Reed, S. Training deep neural networks on noisy labels with bootstrapping., 2014.

[8] Simonyan K. Zisserman-A. Kavukcuoglu K. Jaderberg, M. Spatial transformer networks., 2015.

[9] Van Horn G.-Belongie S. Perona P. Branson, S. Bird species categorization using pose normalized deep convolutional nets., 2014.

[10] Saberian M. Yang-J. Li L.J. Vasconcelos N. Belongie S. Moghimi, M. Boosted convolutional neural networks., 2016.

[11] Zhou F. Wang-J. Liu X. Lin Y. Belongie S. Cui, Y. Kernel pooling for convolutional neural networks., 2017.

[12] Hadsell R. LeCun-Y. Chopra, S. Learning a similarity metric discriminatively, with application to face verification., 2005.

[13] Grauman K. Parikh, D. Relative attributes., 2011.

[14] Sapp B. Howard-A. Zhou H. Toshev A. Duerig T. Philbin J. Fei-Fei L. Krause, J. The unreasonable effectiveness of noisy data for fine-grained recognition., 2016.

[15] Vilnis L. Le-Q.V. Sutskever I. Kaiser L. Kurach K. Martens J. Neelakantan, A. Adding gradient noise improves learning for very deep networks., 2015.

[16] Vanhoucke V. Ioffe-S. Shlens J. Wojna Z. Szegedy, C. Rethinking the inception architecture for computer vision., 2016.

[17] https://cropwatch.unl.edu/plantdisease/soybean/frogeye-leaf-spot, 2021.

[18] https://extension.umn.edu/plant-diseases/apple-scab, 2021.

[19] https://extension.psu.edu/apple-diseases-rust, 2021.

[20] https://extension.psu.edu/apple-disease-powdery-mildew, 2021.

[21] Fredrik Lundh, Alex Clark, and Contributors. Pillow¶.

[22] scikit-learn developers. sklearn.svm.svc¶.

[23] A. Y. Ng A. L. Maas, A. Y. Hannun. Rectifier nonlinearities improve neural network acoustic models., 2013.

[24] G. Sun J. Hu, L. Shen. Squeeze-and-excitation networks, 2018.

[25] R. Socher L.-J. Li K. Li L. Fei-Fei J. Deng, W. Dong. A large-scale hierarchical image database, 2009.

[26] J. Ba D. P. Kingma. Adam: A method for stochastic optimization, 2014.

[27] R. P. Ribeiro P. Branco, L. Torgo. A survey of predictive modelling under imbalanced distributions, 2015.

[28] et. al. Paszke. Pytorch: An imperative style, high-performance deep learning library, 2019.

[29] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition, 2015.

[30] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016.

[31] https://pandas.pydata.org/, 2021.

[32] https://numpy.org/, 2021.

[33] https://www.tensorflow.org/, 2021.

[34] https://keras.io/, 2021.

[35] Laurens van der Maaten Kilian Q. Weinberger Gao Huang, Zhuang Liu. Densely connected convolutional networks, 2016.

[36] Quoc V. Le Mingxing Tan. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019.

[37] Vincent Vanhoucke Alex Alemi Christian Szegedy, Sergey Ioffe. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[38] Sergey Ioffe Jonathon Shlens Zbigniew Wojna Christian Szegedy, Vincent Vanhoucke. Rethinking the inception architecture for computer vision, 2015.

[39] Bo Chen Dmitry Kalenichenko Weijun Wang Tobias Weyand-Marco Andreetto-Hartwig Adam Andrew G. Howard, Menglong Zhu. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[40] Jonathon Shlens Quoc V. Le Barret Zoph, Vijay Vasudevan. Learning transferable architectures for scalable image recognition, 2017.

[41] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale image recognition, 2014.

[42] https://keras.io/api/applications/, 2021.

Author Biographies

Mateusz Chilinski is currently a PhD student in the field of Information and Communication Technology at Warsaw University of Technology. He has obtained his Bachelor of Science in Engineering, and Master of Science in Engineering at Faculty of Mathematics and Information Technology at Warsaw University of Technology, in the field of Computer Science, and Computer Science and Information Systems. Beside his Computer Science-related studies, Mateusz has completed Bachelor of Arts in Management at Kozminski University, and is due to graduate in Master of Arts in Project Management at Warsaw School of Economics in 2021. He is interested in computational approaches in molecular biology. Currently working on relations between 3-dimensional structure of the human genome, genetic variation (concentrating mostly on structural variants), and resulting gene expression. Mateusz is doing his PhD within the project "Spatial network model of sequence and structure diversity of Human genome at a population scale" financed by the PRELUDIUM-BIS program of National Science Centre, Poland.

Piotr Goralewski is a PhD student in the field of Information and Communication Technology at Warsaw University of Technology. He has completed his BSc and MSc studies in the field of Applied Computer Science at Faculty of Electrical Engineering at Warsaw University of Technology. His PhD research is related with Computer Vision and Natural Language Processing. He is also interested in the areas of bioinformatics, genetics and recommender systems.

Tomasz Lehmann is a PhD student at Warsaw University of Technology. He has completed full time studies in field of Applied Physics and obtained his Master of Science in Engineering at Warsaw University of Technology. During his PhD researches he is focused on Computer Vision area and specially on biometric recognitions.

Adam Nowacki is a PhD student in the field of Information and Communication Technology at Warsaw University of Technology. He has completed his BSc and MSc studies in the field of Quantitative Methods in Economics and Information Systems at Warsaw School of Economics. His PhD research is related with Computer Vision and Deep Learning and is mainly focused on satellite imagery. As a Data Scientist, he professionally develop forecast models, put into production optimization algorithms for supply/logistic chain and implement machine learning practical applications.

Justyna Stypulkowska is a PhD student in the field of Information and Communication Technology at Warsaw University of Technology. She has obtained two Bachelor of Science in Engineering degrees: in the field of Computer Science, Electronics and Telecommunications at Faculty of Electronics and Information Technology at Warsaw University of Technology and in the field of Mechanical Engineering at Faculty of Automotive and Construction Machinery Engineering at Warsaw University of Technology. She has obtained the degree of Master of Science in Engineering in the field of Mechatronics, specializing in Multimedia Techniques at Faculty of Mechatronics at Warsaw University of Technology. Her doctoral thesis is related to Computer Vision, the use of deep learning in image recognition and the analysis of the obtained results depending on the adopted hardware solutions. She works as a Programmer and Specialist of Databases, Cloud Solutions and Machine Learning at the Łukasiewicz Research Network - Institute of Aviation. She is interested in new deep learning solutions, Computer Vision and music mastering.