BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE
ÚSTAV AUTOMATIZACE A INFORMATIKY

AUTOMATIC IDENTIFICATION OF TREE PESTS BASED ON IMAGE DATA

AUTOMATICKÁ IDENTIFIKACE ŠKŮDCŮ DŘEVIN NA ZÁKLADĚ OBRAZOVÝCH DAT

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — Ing. Marek Balko
AUTOR PRÁCE

SUPERVISOR — doc. Ing. Pavel Škrabánek, Ph.D.
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Master's Thesis

Institut:	Institute of Automation and Computer Science
Student:	**Ing. Marek Balko**
Degree programm:	Applied Computer Science and Control
Branch:	no specialisation
Supervisor:	**doc. Ing. Pavel Škrabánek, Ph.D.**
Academic year:	2023/24

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

Automatic identification of tree pests based on image data

Brief Description:

Each year, insect pests cause considerable damage to forests, urban greenery, and agricultural trees. Monitoring the occurrence of these pests is an important part of prevention in the management of woody plants. The simplest non–destructive method of the monitoring is visual inspection of the plants. In the case of visual monitoring by a person without instrumentation, the quality of the monitoring is highly dependent on the knowledge and experience of the person carrying out the monitoring. Due to significant advances in computer vision, it is possible to develop tools that allow to general public to carry out the monitoring at the expert level. Such a tool can thus greatly improve prevention in the management of woody plants.

Master's Thesis goals:

The student will prepare a survey mapping technique used to identify pest species based on image data. The student will annotate a specified dataset, design and implement a computer vision system aimed at identification of specified insect pests, and evaluate the effectiveness of the developed system.

Recommended bibliography:

GUO, Ying, Junjia GAO, Xuefeng WANG, et al. Precious Tree Pest Identification with Improved Instance Segmentation Model in Real Complex Natural Environments. Forests [online]. 2022, 13(12) [cit. 2023-09-05]. ISSN 1999-4907. Dostupné z: doi:10.3390/f13122048

RAUM, Susanne, C. Matilda COLLINS, Julie URQUHART, Clive POTTER, Stephan PAULEIT a Monika EGERER. Tree insect pests and pathogens: a global systematic review of their impacts in urban areas. Urban Ecosystems [online]. 2023, 26(2), 587-604 [cit. 2023-09-05]. ISSN 1083-8155. Dostupné z: doi:10.1007/s11252-022-01317-5

SUN, Yu, Xuanxin LIU, Mingshuai YUAN, Lili REN, Jianxin WANG a Zhibo CHEN. Automatic in-trap pest detection using deep learning for pheromone-based Dendroctonus valens monitoring. Biosystems Engineering [online]. 2018, 176, 140-150 [cit. 2023-09-05]. ISSN 15375110. Dostupné z: doi:10.1016/j.biosystemseng.2018.10.012

LI, Wenyong, Tengfei ZHENG, Zhankui YANG, Ming LI, Chuanheng SUN a Xinting YANG. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. Ecological Informatics [online]. 2021, 66 [cit. 2023-09-05]. ISSN 15749541. Dostupné z: doi:10.1016/j.ecoinf.2021.101460

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2023/24

In Brno,

L. S.

Ing. Pavel Heriban, Ph.D. Director of the Institute	doc. Ing. Jiří Hlinka, Ph.D. FME dean

ABSTRACT

This thesis focuses on utilizing image data of tree trunk damage to train a classifier for recognizing species of tree pests that caused this damage. The classifier is designed as a convolutional neural network. To successfully train the model, a preprocessing step - the sub-image generator - was employed before the classifier. This generator creates training data of suitable dimensions by cropping from the original data. The resulting data retains important details for network training. Two methods for generating training sub-images were proposed for the sub-image generator - the Grid division method and the Elliptic division method. Both of these methods can be successfully used to train the classifier for tree pest recognition based on image data of tree damage with comparable model accuracy. The Elliptic division method is more flexible and less time-consuming for preprocessing training data.

ABSTRAKT

Tato diplomová práce se věnuje využití obrazových dat poškození kmene stromu k natrénování klasifikátoru pro rozpoznávání druhů škůdců stromů, které toto poškození způsobili. Klasifikátor je navrhnut jako konvoluční neuronová síť. Pro úspěšné natrénování modelu byl klasifikátoru předřazen preprocesingový krok – sub-image generátor. Tento generátor vytváří tréninková data o vhodných rozměrech pomocí výřezů z původních dat. Takto vzniklá data zachovávají důležité detaily pro trénování sítě. Pro sub-image generátor byly navrženy dvě metody vytváření trénovacích pod-obrazů – Grid division method a Elliptic division method. Obě tyto metody lze úspěšně použít pro natrénování klasifikátoru škůdců stromů na základě obrazových dat poškození stromu se srovnatelnou přesností modelu. Metoda Elliptic division je flexibilnější a méně časově náročná na preprocesing trénovacích dat.

KEYWORDS

CNN, classifier, tree pests, Grid division method, Elliptic division method

ÚSTAV AUTOMATIZACE A INFORMATIKY

2024

BIBLIOGRAPHIC CITATION

BALKO, Marek. *Automatic identification of tree pests based on image data*. Brno, 2024. Available from: https://www.vut.cz/studenti/zav-prace/detail/157605. Master's Thesis. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Supervised by Ing. Pavel Škrabánek, Ph.D

ACKNOWLEDGEMENT

I would like to thank Doc. Ing. Pavel Škrabánek, Ph.D., for his professional approach and valuable advice during the supervision of my master's thesis. Additionally, I would like to express my gratitude to Ing. Christo Nikolov, Ph.D., and Ing. Milan Zúbrik, PhD., from National Forest Centre, Forest Research Institute, Slovakia, for providing the database of tree pest photos.

AUTHOR'S DECLARATION

I declare that this thesis is my original work, I have prepared it independently under the supervision of the thesis supervisor and using professional literature and other sources of information, all of which are cited in the thesis and listed in the reference list.

As the author of this thesis, I further declare that in connection with the creation of this thesis I have not infringed the copyright of third parties, in particular I have not infringed in an unauthorised manner on the personal copyright of others, and I am fully aware of the consequences of violating the provisions of § 11 et seq. of the Copyright Act No. 121/2000 Coll., including possible criminal consequences.

In Brno, 20. 5. 2024 .....

Marek Balko

CONTENTS

1 INTRODUCTION

Trees and forests have been a crucial environmental factor for humans since immemorial. Previously, they served primarily as a source of food, shelter, and construction materials. In the present era, we can identify a myriad of effects that forests have on our planet and human society.

Forests are directly linked to water and the water cycle. They contribute to water retention within themselves and the surrounding landscape. They act as filters for rainfall, improving the replenishment of groundwater. Forests also enhance air moisture and can directly influence precipitation [1].

One of the most significant effects of all green plants is their ability to perform photosynthesis, wherein they can convert carbon dioxide into oxygen, essential for all living organisms, in the presence of sunlight. Given that forests cover approximately one-third of the Earth's land surface, their contribution to oxygen production is significant and very important for all living species. Additionally, trees capture large volumes of carbon, filtering it from the atmosphere, thereby reducing the impact of carbon dioxide as a greenhouse gas [1].

Another important aspect of trees and forests is their ability to cool their surroundings through the water stored in them. Exploiting this phenomenon makes sense, especially in urban areas and cities, where temperatures on concrete and metal surfaces can become unbearable, particularly during summer. Urban forests can help mitigate the heat stress in their vicinity. They also create spaces for relaxation and recreation for residents in the area, thereby improving the social status of the given location [2].

Forests also serve as habitats for a wide range of indigenous species of animals and plants specific to a particular locality, thus creating self-sustaining ecosystems. In addition to all the aspects mentioned above, it is essential to acknowledge that forests hold economic significance for humans, serving as a source of timber, which continues to have a broad range of applications in various industrial sectors.

All these factors underscore the need to ensure the quality, health, and quantity of forests in our landscape. Uncontrolled deforestation, whether natural or artificial, can have catastrophic consequences. Dying trees release greenhouse gases into the air, reduce oxygen production, and contribute to the reduction of the transformation of carbon dioxide. The landscape experiences drying out and an increase in ambient temperature, potentially leading to droughts. Without sufficient land cover from the root systems of trees, the area becomes more susceptible to floods and soil erosion. The extinction of certain species of animals or plants may occur if their natural habitat diminishes or disappears entirely. It is evident from this that caring for the forests on our planet is crucial because they constitute an integral part of the entire ecosystem, of which we humans are a part [3].

Although damaged, dead, or harvested trees can be replaced through reforestation - planting new trees in areas where they once existed, or afforestation - establishing forests in entirely new locations where there was no forest before, it is crucial to proactively care for the quality and health of forests and prevent more serious scenarios that could have significant economic and environmental impacts. Basic threats to forests include fires, droughts, ice storms, climate change, monoculture planting, fungi, diseases, and pests [3]. This work will focus on the last mentioned - tree pests.

Tree pests represent a significant threat to forest environments. They attack and damage trees which can even lead to their death. Non-native species often pose the most significant problem among pests, as local ecosystems may not have developed robust defences against them. These species frequently lack natural predators in the local environment, leading to uncontrolled population growth. Due to globalization and easier international trade between distant states, pests can easily spread to areas where they are not a natural part of the environment, causing problems. Warmer and drier conditions also typically contribute to the accelerated growth of pest numbers. Tree pests decimating forest stands have a considerable negative impact on the economy, ecology, and the environment [4].

The fundamental idea of traditional forest management regarding pests involves the removal of dead and damaged trees, known as salvage logging. The goal of this method is to eliminate such trees and thereby slow down the spread of pests by depriving them of places for further breeding. This approach also includes the removal of fallen trees due to natural events such as strong winds, storms, or landslides because these dead trees become easy targets for pests. Although salvage logging is a widespread method, solid scientific evidence for its effectiveness remains doubtful [5].

Another common method is sanitation logging. This involves a swift intervention in the center of infection, where infected trees are precisely removed before pests can rapidly spread to the surrounding area and cause more serious problems. The timing of sanitation logging is a crucial aspect of the method, as the window for intervention is typically short after the detection of symptoms and the spread of adult pests to other trees. In this case, a well-established forest monitoring system is important. There is a need for sufficient manpower with the necessary skills and knowledge for detecting various types of tree damage, evaluating and classifying the damage, determining whether it is caused by pests, classifying the specific pest, and deciding whether it poses a problem or not. Such requirements are both financially and time-intensive [5]. Automating this process makes sense to alleviate the burden on human workers. Particularly, there is an opportunity to attempt to create an automated pest classifier based on image data of tree damage. Such a tool could significantly reduce the qualifications needed for field monitoring staff and, consequently, reduce time and financial costs. The primary assessment of the feasibility of such a tool is the goal of this master's thesis.

As briefly demonstrated, forests are a crucial element of our planet, and their existence has a direct impact on humanity. This leads to a clear goal of caring for the

quality and health of forests, which can be threatened by various influences, whether natural or artificial.

This master's thesis aims to explore the feasibility of developing a classification tool capable of identifying tree pests based on image data of tree damage, which could be directly applicable in the field. This work is being conducted in collaboration with a National Forest Centre, Forest Research Institute, Slovakia, which provided input databases of labelled photographs and expressed interest in the development of such a classifier. The introductory part of the thesis maps the state of the art of possibilities for classifying tree pests based on image data. The practical section then focuses on the methodology and implementation of evaluating initial possibilities for pest classification using convolutional neural networks with input images of tree damage.

2 STATE OF THE ART IN IMAGE CLASSIFICATION FOR PEST MONITORING

Image classification is a machine learning task and is a fundamental aspect of computer vision, where the goal is to assign pre-defined classes to specific images. Each image can be assigned one or more classes. Image classification typically falls under the category of supervised learning. The training data set must be labelled with tags or labels, serving to train the model so that it can classify new images in the future that were not shown during the model's training.

In contemporary articles, pest classification is often also associated with their detection. Object detection is also a machine learning and computer vision task. In contrast to image classification, where the goal is to assign a single class to the entire image, object detection aims to identify multiple objects in a given image and correctly classify each of them. The output includes the classes and positions of individual objects in the image. Again, this is a type of supervised learning task, where labelled training data is required for training the model [6].

The decision to use image classification or detection depends on the specific problem requirements. For example, if we need to monitor whether a particular pest species is present on a plant, image classification alone may suffice. However, if we need to determine not only the species but also the number of individuals of specific pests on the plant, image detection is required. This may be necessary, for example, for pest species where low-level occurrence is common, and the danger arises only from their proliferation, which can be monitored in this way. Practical implementation of both tasks can be achieved using a variety of machine learning and deep learning methods. Selected methods will be introduced in the chapter 3 of this work, considering their current most common use for pest detection and classification.

Fig. 1 shows the scheme of the most commonly used methods and their dependencies on input data. The selection of a suitable method also varies based on the data we want to use for training the model. Each method requires a different type of input from the original image data, which affects the necessary degree of image preprocessing before creating the classification or detection model. Currently, among classical machine learning methods, Support Vector Machines are almost exclusively used [7]. The state-of-the-art in image recognition and detection involves the use of deep learning, specifically convolutional neural networks. It is shown that this approach outperforms classical machine learning methods. It also reduces the requirements for model creation in the sense that it is able to work with original images without the need for its further extensive processing [7].

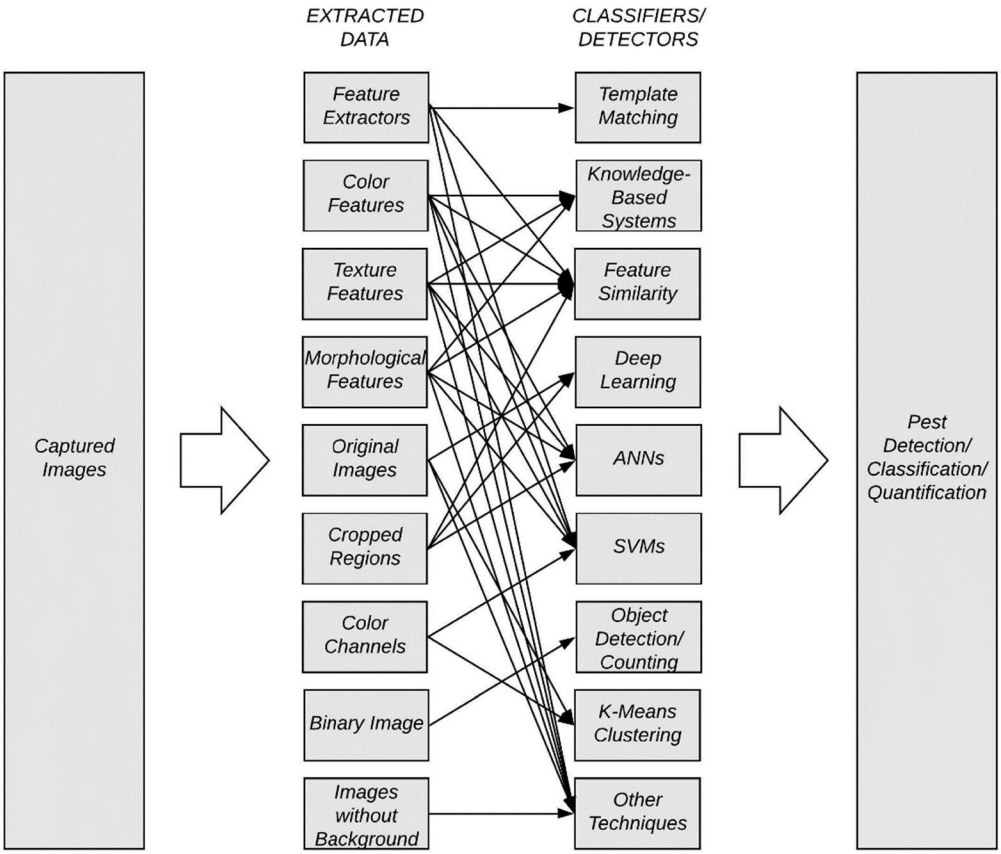

Fig. 1 - Commonly used ML methods and their dependencies on input data [7]

For example, the work [8] employs the SVM method for detecting and classifying pests on strawberry flowers. Image data were collected automatically using a robot that travels above pots with strawberry plants and takes photographs of the flowers. For successful deployment of the Support Vector Machines method, the authors had to employ strong data preprocessing and subsequent image feature extraction. Primarily, this involved classic operations such as brightness and geometric transformations, as well as morphological operations like dilation and erosion. The entire sequence of operations aimed to extract from the original photo an image of the flower with the pest without the background and other unwanted parts. Subsequently, several features were extracted from such an image, including geometric features such as the region index (the ratio of the flower's radius to the pest's radius) and colour features like hue, saturation, and intensity calculated from the RGB values of the image. These features were used as input to the SVM with radial basis function kernel. Their model was capable of detecting whether targeted pests were present on the flower or classifying the type of insect (from 5 classes) with an error of less than 3% [8].

The work [9] collects image data using light traps and focuses on the classification and counting of individuals of various pest species on rice plants. Insects caught in the light trap were prepared once a day for classification and counting. Initially, it was necessary to pour the captured insects onto a glass plate placed in a holder so that individual specimens did not touch each other, and to manually remove insect species that were not the target of classification. The entire scene was captured by two cameras

and four light sources. Photos from both cameras were used for geometric and brightness transformations and subsequent segmentation. After data preprocessing, 156 features containing information about geometry, colour, hue saturation value, texture, and others were extracted from the adjusted photos. These features were input into the training of an SVM model with a radial basis function kernel. The authors' model was capable of achieving a classification accuracy of 97,5% [9].

Although both models exhibit high accuracy, their usability is very limited. The support vector machines method requires specific inputs for its proper functionality. In the case of the second work, the scene had to be created manually, which greatly limits the method's usability. Furthermore, the necessity of strong preprocessing of image data requires a similar scene in each photo and may also result in the loss of some information, for example, if pests of different sizes appear in the photo, the smaller ones can be erased during transformations. The high accuracy of the models is thus achieved at the cost of increased time, effort, and personnel required for data preprocessing, which reduces the applicability of this method, for example, for online pest monitoring. However, it is still worth noting that despite its demanding input data requirements, the Support Vector Machines method is the most usable among classical machine learning methods for pest detection and classification.

In the field of monitoring, classification, and detection of insects, the use of convolutional neural networks is the most widely used state-of-the-art method. Due to their simplicity in input data preprocessing and their ability to extract comprehensive features from the original data, CNNs are increasingly becoming the first choice for developing applications for pest monitoring [10]. Specific CNN architectures vary depending on the problem being addressed. CNNs differ in architecture, number of classification classes, usage, resulting accuracy, and more. Comprehensive reviews on this issue can be found, for example, in [10]. Generally, summarizing the use of CNNs for pest detection and classification is challenging because each model varies significantly from others in many aspects. However, it can be stated that convolutional neural networks are capable of performing more complex tasks compared to classical machine learning methods. The main disadvantage of CNNs is the requirement for a large amount of training data, where the general rule is the more data per class, the better. Recommended values for training networks typically start in the range of thousands of images per class, and obtaining such a quantity of data may not be easy or feasible. The quality of input data directly affects the trained model.

3 COMPUTER AIDED MONITORING OF TREE PESTS

3.1 Data acquisition

For any machine learning method, it is important to have high-quality input data to create a model. The way data is collected can influence the final model's performance. In the field of pest classification and detection, several basic methods are used to obtain input image data. Each of these methods has its advantages and disadvantages.

The first and most common method is the use of pheromone or light traps. These are typically hanging devices placed in forests or agricultural areas, attracting various types of insects that become trapped inside. Typically, light traps can attract more diverse insect species simultaneously compared to pheromone traps, which attract a lower number of insect species. The use of both types of traps is straightforward [7].

Subsequent acquisition of image data typically leads to one of two methods. The most common method involves manually collecting traps at regular intervals. The captured insects are then transferred to a laboratory where they are randomly sifted onto an illuminated surface, manually separated, and then photographed by a camera fixed to the board. The advantage of this method is that the resulting photos always have the same lighting conditions, minimizing overlap between individual insect specimens, thus creating relatively high-quality input data. The disadvantage is the labor-intensive process of manually collecting traps and subsequent separation and preparation of samples. Additionally, this method involves a significant time delay between insect capture and final sample evaluation, which may be a significant negative factor, especially for monitoring pest outbreaks [7; 10].

The second method of obtaining input photos involves using more complex traps equipped with lighting and a camera inside. Images of the inside of the traps with captured insects are typically taken at certain time intervals. Such photos can then be easily sent to a remote device for further processing. This type of trap is more expensive, and the resulting photographs may have slightly lower quality due to the potential overlap of captured insects in the photos [11].

The use of both types of traps and methods of collecting image data shares a common disadvantage. These traps are only capable of capturing the insects or stages of insect development that are capable of flying. However, often the larval stages, which are incapable of flight, are more dangerous to crops and trees, and thus their monitoring could be more important.

Another commonly used method for generating input data is the utilization of coloured sticky traps. Different types of insects are attracted to different colours. Insects become stuck to the adhesive surface of the tape, which is placed in suitable locations such as greenhouses, forests, fields, etc. These tapes are then collected and subsequently

photographed [12]. Essentially, this method is very similar to the method of using light or pheromone traps with manual collection, sharing both the common advantages (simplicity, good data quality) and disadvantages (labor-intensive, suitable only for flying insects) [13].

The last and less common method is acquiring input data directly in the field. This can be easily achieved using any camera, typically a smartphone, that a person may have with them. Insect photos are taken directly in their natural environment without any external intervention [14]. In some cases, this data collection can be automated using mobile robots equipped with cameras. This approach can be commonly used in greenhouses, where plants, on which pests may reside, are regularly and definedly distributed in space [8].

The advantage of this method of obtaining input image data is that it provides data that could realistically serve as input for a future application, rather than just laboratory simplification, which leads to higher quality input data but at the cost of reducing realism. The disadvantages stem from the nature of the method, such as differing lighting and shadows in each photo, irregular and complex backgrounds, the presence of other objects that are not the objects of interest. However, this method is capable of working with any type of insect, not just flying species.

With the increasing computational capacity of computers and the development of convolutional neural networks, the use of this method appears more justified. Assuming a sufficiently large number of input data for training, the disadvantages of this method may become its advantages because a complex model will be able to sufficiently generalize the nature of the input data, making its deployment in practice easier and more useful.

In general, there is no one best method for collecting input data, and it is necessary to always consider the specific problem being addressed. In addition to the aforementioned issues in data collection for classification and detection, it is also necessary to mention that when using any method, the input images may contain many other types of insects that may not be pests (such as bees, ants, etc.), but can still negatively affect the future functionality of the model. The varying sizes of individual insect species can also have a similar impact. Currently, the most commonly used method is the use of light or pheromone traps, but an increase in the use of data from real environments can be expected.

3.2 Automatic identification of tree pests using image data

In the literature, articles focusing on pests in agriculture prevail, while pests in forestry constitute a significantly smaller portion. Almost exclusively, image data containing individual pests are used for classification problems, providing an opportunity to explore the possibility of classifying pests based on image data containing plant damage rather than the insects themselves. Deep learning, specifically convolutional

neural networks, is the most used method for automatic identification with small except for Support Vector Machine method.

In the context of machine learning, there is no actual research focused on using pictures of damaged trees as input data for training classification models. Such an approach could lead to interesting applications and easier acquisition of input data, as it is not always easy to find the pest responsible for the damage on an affected crop or tree. It could also involve a dead damaged tree where no pests are present anymore, yet there could still be a possibility to identify which pest caused the death of the tree. Overall, the use of images of plant damage as input data could lead to new application possibilities for monitoring crop or tree pests. This work aims to explore the possibility of using photos of tree damage for the classification of the pest responsible for the damage using convolutional neural networks.

4 IMAGE CLASSIFICATION

4.1 Machine learning

Machine learning can be defined as the process of solving a specific practical problem by collecting a training dataset and constructing a statistical model algorithmically based on this data. With such a model, we assume it is capable of addressing our defined problem. Machine learning is hierarchically categorized into tasks of artificial intelligence, where artificial intelligence itself is capable of learning new approaches how to achieve goals, based on which it performs tasks. One of the subcategories of machine learning is deep learning, where the artificial intelligence model is capable of learning in a more complex way from data. The scheme of this hierarchy is commonly depicted in Fig. 2. Typical machine learning problems include classification, detection, regression, prediction, clustering, or dimensionality reduction [15].

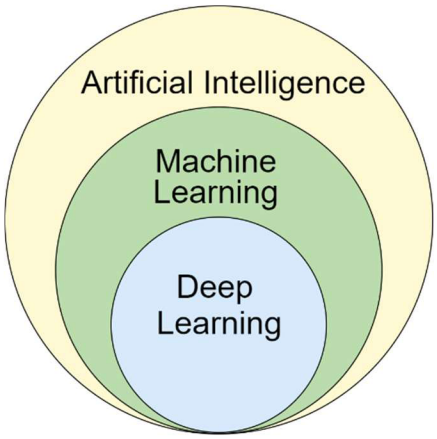

Fig. 2 - Artificial intelligence division hierarchy [16]

Based on the dataset, we can distinguish several basic ways of training statistical models, including supervised learning, unsupervised learning, and reinforced learning. The selection of algorithms for building a statistical model determines the specific method used. Among the classical machine learning methods are Bayesian networks, K-means clustering, decision trees, artificial neural networks, or Support vector machines [17].

In addition to classical machine learning methods, there are more specialized methods falling under the category of deep learning, which work with deep neural networks. Here, the widely used method of convolutional neural networks will be introduced, which is very suitable for working with multidimensional data, such as images. Convolutional neural networks are generally widely used for detection and classification in image data, thus being heavily employed for working with photos of pests on plants [15].

In supervised learning, the dataset consists of pairs of known data x and y, where x is a feature vector of some dimensionality D, and y is a label determining the assignment of the given pair. The elements of the label y can belong to a finite set of classes or can

be a real number or a more complex structure such as vectors, matrices, trees, or graphs. The feature vector x contains all the relevant information for assigning the correct label. All data points x in the dataset must have the same order of features (individual elements of the vector) throughout the feature vector [15].

In supervised learning algorithms, feature vectors x and their corresponding output labels y are gradually passed to the model. The model then learns to deduce the corresponding output y for the presented input feature vector x.

In unsupervised learning, the dataset consists of unlabelled feature vectors x. The goal of this learning algorithm is to create a model that takes an input vector x and either transforms it into another vector or into a specific value that addresses a particular problem for which the model is designed. For example, clustering is a common problem where the model assigns a cluster label to the input vector x, or dimensionality reduction, where the output is a feature vector of smaller dimensionality than the input feature vector x [15].

Reinforcement Learning differs the most from the previous methods. Here, we work with a model that exists in an environment and is able to observe the state of this environment. The output of such observation is then a feature vector x. The model can take action in each state. Different actions yield different rewards and may move the model to another state in the surrounding environment. The goal of a Reinforcement Learning algorithm is to learn such a policy that, given an input observation vector (feature vector x), outputs the optimal action to take in that state, maximizing the expected average reward obtained from performing the action. Reinforcement Learning is used to solve specific types of problems involving sequential decision-making and long-term goals. Such tasks may include playing games, robotics, resource management, or logistics [15].

4.1.1 Support Vector Machines

The Support Vector Machines is mostly used method for classifying images of pest from classical machine learning. The main idea of Support Vector Machines (SVM) is to find a hyperplane (a line in 2D space) that completely separates different classes of data while maximizing the distance from both neighbouring classes. The dimensionality of the space in which data separation is sought is determined by the number of elements in the feature vector x. The assignment to data classes is represented by a label vector y. Assuming that the classes are linearly separable, finding such a hyperplane is straightforward, and its orientation depends only on the closest data points, known as support vectors. This fact significantly reduces the computational complexity of training and classification. The hyperplane for 2-D case with two classes is demonstrated on Fig. 3 [18].

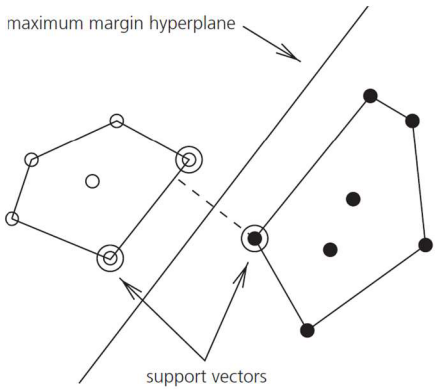

Fig. 3 - Linear separable data [18]

The assumption of linear separability of data is very limiting and prevents broader use of the method on noisy data or data that is separable but not linearly. Fortunately, both of these problems can be addressed. To work with noise, the loss function can be adjusted to place more or less emphasis on noisy data. This emphasis is controlled by a hyperparameter that balances the trade-off between good classification of training data (greater emphasis on noisy data) and good generalization of features for future data samples (less emphasis on noisy data) [18].

The Support Vector Machine method can be adapted to work with datasets that cannot be separated by a hyperplane in their original space. This can be achieved by transforming the original space into a higher-dimensional space, where it will be possible to find a hyperplane that can linearly separate the two classes. An example of such transformation is shown in Fig. 4. It can be seen, that in picture *a*, it's possible to separate both cases, but not linearly. By transforming from two-dimensional space into three-dimensional space in picture *b*, it's now possible to create plane that linearly separate both classes. The mapping of the original space values to the higher-dimensional space is done using the so-called kernel trick. Various kernels can be used for this transformation, with the Radial Basis Function (RBF) kernel being the most commonly used one, whose feature space has an infinite number of dimensions [18].

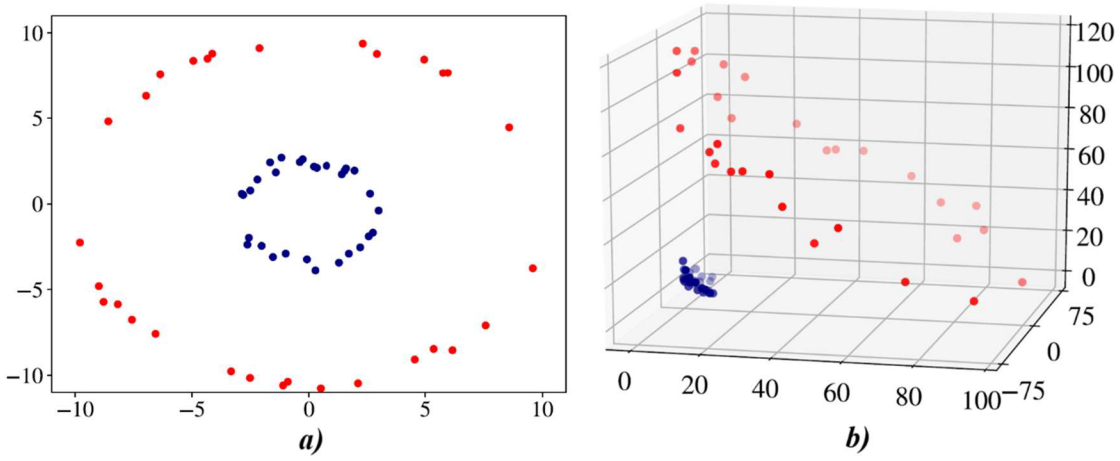

Fig. 4 - a) Nonlinear separable data, b) Transformed data [15]

A crucial component of Support Vector Machines is the feature extractor. It's a step in the algorithm that transforms input image data into input feature vectors x. Its aim is to summarize the image's properties into user-defined numerical values - features, which can further be used for model training. This may include values such as hue, saturation, or intensity computed from the RGB image values and many others. The feature extractor is a vital part of the SVM algorithm, and selecting appropriate features strongly influences the final ability of the model to correctly classify input data.

The aim of this subsection was to briefly introduce the basic ideas of this commonly used machine learning method for pest classification. Its detailed derivation, training algorithms and the use of kernel functions are not necessary for the rest of this thesis, therefor they will not be described into further depth. For those interested, more information can be found in [15; 17; 18].

4.2 Deep Learning

Deep learning is a subcategory of artificial intelligence falling under machine learning. Its popularity began to rapidly increase around 2006, and it is now a widely used approach for handling complex tasks in computer science. Deep learning operates with deep neural networks, which are networks that have a "large" number of hidden layers. However, the definition of a "large number of layers" varies in the literature, and deep neural networks are often considered to be networks with more than one hidden layer of neurons [19].

One significant advantage compared to classical machine learning approaches is its greater ability to generalize and perform more complex reasoning based on simpler data. Consequently, it reduces the demand for high-quality training data inputs because deep networks can extract important features from more simple data themselves. On the other hand, to train models optimally, a large amount of training data is required, which can easily range from tens of thousands to millions of samples for each class[19].

Generally, models based on deep learning outperform classical machine learning models. For the classification and detection in image data (and generally for working with multidimensional data), convolutional neural networks (CNN) are most commonly used. Their functioning will be described in more detail here because they are a fundamental method for this thesis. Other methods of deep learning include, for example, Deep Neural Networks (DNN), Recurrent Neural Networks (RNNs), Long Short-Term Memory models (LSTM), or Generative Adversarial Networks (GAN) [19].

4.2.1 Convolutional Neural Networks

Convolutional Neural Networks are neural networks that contain at least one convolutional layer. Convolution, in general, is a mathematical operation on two functions with real-valued arguments. It can be defined for any number of dimensions, but it is commonly used in two dimensions and on discrete data, such as images. The mathematical formula for 2D discrete convolution is given as [19].

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (1)$$

where S is the result after convolution at pixels i and j. Matrix I is the original image, and matrix K is the kernel. The asterisk operation denotes convolution. Variables m and n, like i and j, are coordinates in the matrix. Convolution is a commutative operation.

The operation of convolution on image data can be imagined as applying a kernel K to the original image, multiplying overlapping values, and summing them all up. This results in a single scalar value, which is the output of the convolution. Subsequently, the kernel is shifted by one position, and the entire process is repeated until the kernel passes through all valid positions. The principle of 2D convolution is shown in Fig. 5. The values in the kernel matrix cause different behaviours of the convolution results. For example, convolution can be used for data averaging in the neighbourhood - smoothing the image, edge detection or many others.

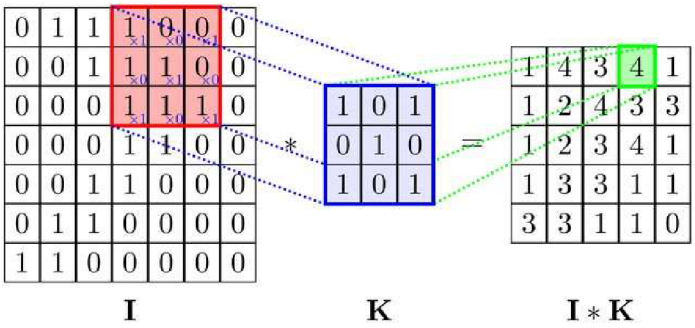

Fig. 5 - Principle of convolution [20]

The values of the elements in the kernel matrix act as parameters that are adjusted through training the neural network, allowing the network to adapt to the optimal features it needs to extract from the input in the convolutional layer. By its nature, the use of convolution reduces the spatial dimension of the output data compared to the input data, depending on the size of the kernel matrix. If this effect is undesirable, zero padding is applied, where the edges of the input data are virtually supplemented with zeros (increasing the spatial dimension) so that the reduced dimension of the output data corresponds to the dimension of the input data before zero padding [19].

Generally, within one convolutional layer, not only one kernel is used but a whole series of kernels. Thus, many convolutions are performed, extracting a variety of features from the input data within one layer. It is common for a CNN to have multiple convolutional layers at different depths of network depending on the specific architecture of the particular network.

Compared to traditional fully connected layers, convolution has a huge advantage in that it requires significantly fewer parameters to perform the extraction of the same feature. Within a kernel, there is one set of parameters used to compute the output from all inputs, whereas a fully connected layer has a specific parameter for each input-output connection. Thus, the use of convolution reduces the overall number of model parameters,

thereby reducing memory requirements and significantly positively affecting the speed of network training [19].

Convolutional Neural Networks also typically include Pooling layers, which are often paired with convolutional layers. Pooling layers replace the output of a portion of the network with a statistical summary of its nearby surroundings pixels. Typically, this can involve operations such as max pooling, which selects the highest input value from the given neighbourhood of pixels and outputs it, average pooling, which outputs the average value of the input pixels in the neighbourhood, or weighted average, which performs weighted averaging depending on the distance from the central pixel of the neighbourhood. This statistical summarization results in the reduction of the spatial dimension of the task, simplifying the problem depending on the size of the used neighbourhood around one pixel. Consequently, memory requirements for storing model parameters are reduced.

The principle of pooling is illustrated in Fig. 6. Pooling spaces may or may not overlap, depending on the size of the utilized neighbourhood and the stride parameter, which determines by how many pixels the next pooling window should be shifted. Since pooling operates only within a small defined area, parallel operations of pooling are used to cover the entire input space of the layer. The main advantage and reason for using pooling is its property of making the model approximately invariant to small local shifts in inputs. For example, if a previous convolutional layer detects an edge, the model is able to understand that it is the same edge even if this edge is shifted by a few pixels in different input data, thanks to the pooling layer. In general, it can be said that invariance to local shifts is important when we need to determine whether a feature is present in the input data or not. The specific position of the feature is therefore not crucial [19; 21].

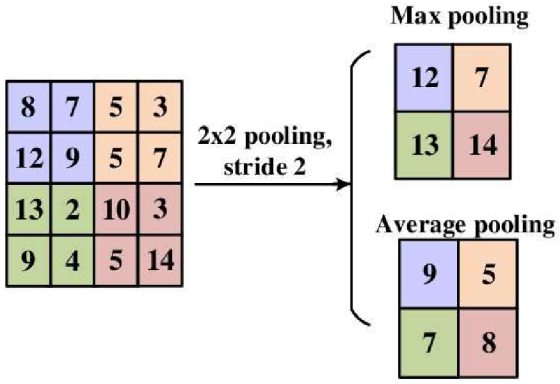

Fig. 6 - Principle of Pooling operation [22]

In Fig. 7, a general diagram of a convolutional neural network for classifying image data is shown. The input is an image with *m* x *n* pixels (width x height of the image). For photos, it is usually assumed that they are coloured, so each pixel carries information about the RGB colour value, thus the input has dimensions of *m* x *n* x 3. CNNs can also be used for grayscale photos or multispectral data, which results in a change in the number of colour channels in the data. The input data are sent to the first convolutional layer.

Depending on each specific model, convolutions are performed separately on each colour channel, and then the results for each pixel are either summed or kept separately for each channel.

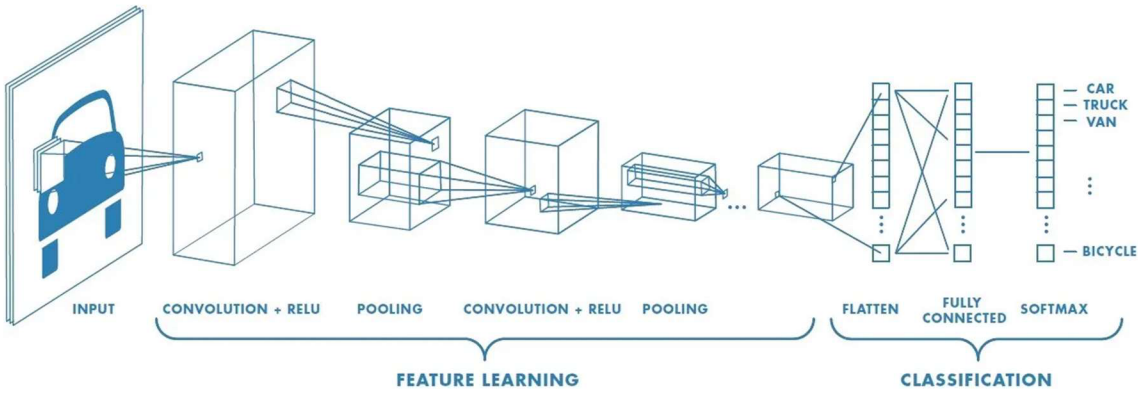

Fig. 7 - General scheme of Convolutional neural network [21]

Convolution is a linear operation, and for the model to learn more complex feature extraction, it is suitable for the neural network to exhibit nonlinear behaviour. This is achieved by introducing activation functions after the convolution output. Typically, the ReLU (Rectified Linear Unit) activation function is used for its simplicity, described by [15]

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise} \end{cases} \qquad (2)$$

where ReLU(x) is the output of ReLU function and x is variable. Graphical interpretations of ReLU function is presented in Fig. 8.

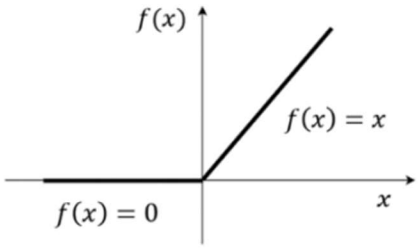

Fig. 8 - ReLU function

After convolution and the ReLU activation function, the data enter the pooling layer. The most common is to use max pooling, which exhibits better behaviour and also serves as a partial noise filter. The pooling layer also reduces the dimensionality of the data. Some architectures include a sequence of several convolutional layers with ReLU functions in succession before their outputs are fed into the pooling layer. The combination of convolutional layers with ReLU and pooling layers is generally repeated several times in succession, creating the main functional blocks of convolutional neural networks. After these blocks, a flattening layer is used, which takes the input 2D matrix and converts it into a 1D column vector. At this stage, the input 2D matrix has a

significantly lower dimension than the original image matrix entering the model's input layer. The outputs from the column vector are then connected to a fully connected layer known from classical neural networks. In this layer, the model learns the connection weights of specific neurons representing the influence of specific complex features extracted from the convolutional parts of the network on classification. A softmax function is then applied to the fully connected layer, which converts the output values of neurons into a probability distribution depending on how many classification classes the model is supposed to recognize. The output from this function can thus be understood as the probability with which the input data belong to each of the classes [19; 21].

4.3 Transfer learning

Training a new neural network is a time-consuming process that requires a vast amount of input data. The complexity of this training process can be reduced using the Transfer Learning method. This method involves taking an already well-trained network on a large amount of data and retraining this network for a different purpose. This process can be imagined as the original network already having learned to recognize complex patterns and features that we would like to use for a new network, but we would like to assign them a different interpretation. For example, various edges or colour elements may be present in any images, but their combination determines the object present in the image.

Typically, several last layers of neural network responsible for classification are removed from the original network and replaced with new layers serving for the classification of new classes. Subsequently, the model is presented with new training pairs of input data, from which the model learns the last minor changes in its parameters to be able to classify new classes. Often, at this stage, some parameters of the original model are frozen, typically in the first layers representing the most basic feature extraction, which is not allowed to change during transfer learning. The largest changes in parameters during learning thus occur in the last layers, especially in those newly added. This commonly used approach reduces the time required to train the full model from scratch. For transfer learning, generally, a lower number of new labelled input data is sufficient compared to the case of a new model. However, the classic rule of deep learning still applies here - the more training data, the better [15; 19].

4.4 Pretrained networks

The use of transfer learning requires the utilization of an existing pre-trained network. There are numerous such networks available, differing in architecture, size, accuracy, speed, or the data on which they were trained. A common scenario is networks trained on the ImageNet database. For training, more than a million different images are used, classified into 1000 different classes corresponding to everyday objects around us (keyboard, cup, pencil, various animals, etc.)[23]. Fig. 9 illustrates accuracy, speed and

space complexity of various pre-trained convolutional neural. Accuracy is referenced to the validation set of images from the ImageNet database, and speed is relative to the fastest network. The size of the points indicates the network's memory requirements.

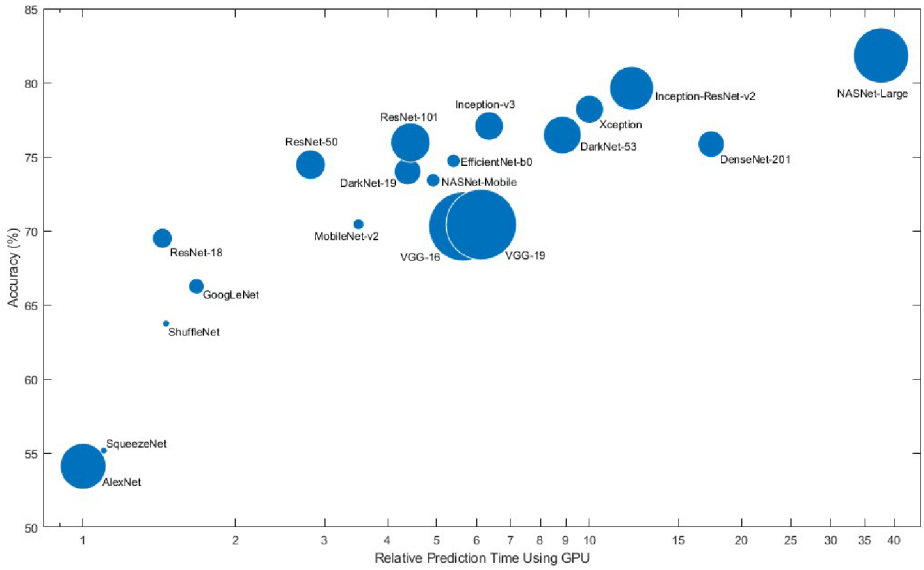

Fig. 9 - Diagram of pretrained CNN [23]

When selecting a suitable network, it is necessary to consider various factors corresponding to the future purpose of the new network. For example, for real-time applications, speed will be an important parameter, while for mobile applications, the network size may be a limiting factor. Generally, there is always a trade-off between size, speed, and accuracy, and the planned purpose of the network must always be taken into account when choosing. Additionally, accuracy is specifically referenced to data from the ImageNet database, on other data, networks may achieve different accuracies.

5 TREE PEST CLASSIFIER

The aim of this work is to investigate the possibility of using image data of tree damage for the classification of tree pests causing this damage. This chapter describes the process of creating three different classification methods for this task, their principles, advantages, and disadvantages.

5.1 Input data

The input data was provided by specialists Christo Nikolov and Milan Zúbrik from National Forest Centre, Forest Research Institute, Slovakia, and consists of data from the database of pest photos www.skodcoviadrevin.sk [24]. This database contains various photos of damage to different parts of trees and shrubs from various causes - diseases, insect pests, fungi, or damage from larger animals. The first part of the practical section of the work thus focused on selecting suitable input data. Firstly, it was necessary to filter the photo database only for tree damage caused by pests. Subsequently, all available records in the database were thoroughly manually checked. For each pest, the number of photos of damage caused by that specific pest available in the database was recorded, along with the location on the tree where the damage occurs - leaves, branches, or trunk. For some species, only photos of insects were available, not the damage they cause, so such pest species were not suitable for this work. Over 380 records of various pest species were reviewed in this manner. Subsequently, 97 species were selected from this list, for which the number of available photos was greater than 15, which then underwent further selection based on which type of damage contained the most photos. Based on these criteria, it was decided that photos of trunk damage from those species with a larger number of photos available (70 or more) would be used for this work. This was followed by discussions with experts on tree pest issues regarding the appropriate selection of species that makes the most sense. From this discussion, the final specification of the selected species emerged. Various representatives of bark beetles typical for Czech and Slovak forests were chosen as training pest species. Specifically, the following seven species were selected: *Cerambyx cerdo*, *Ips acuminatus*, *Ips cembrae*, *Ips sexdentatus*, *Ips typographus*, *Pityogenes chalcographus*, and *Tomicus minor*. Samples of the typical tree damage from selected pests are presented on Fig. 10 - Fig. 16. Tab. 1 displays the counts of photos available for each class.

Fig. 10 - *Ceramryx cerdo* tree damage [24]

Fig. 11 - *Ips acuminatus* tree damage [24]

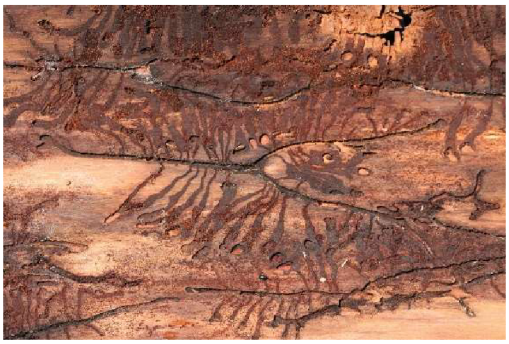

Fig. 12 - *Ips cembrae* tree damage [24]

Fig. 13 - *Ips sexdentatus* tree damage [24]

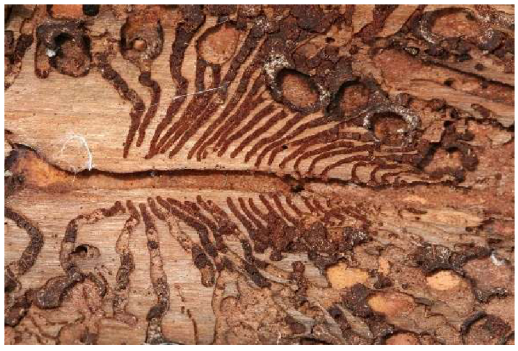

Fig. 14 - *Ips typographus* tree damage [24]

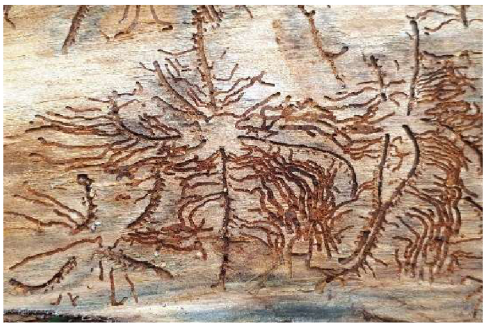

Fig. 15 - *Pityogenes chalcographus* tree damage [24]

Fig. 16 - *Tomicus minor* tree damage [24]

The photos of the last-mentioned species, *Tomicus minor*, were provided by experts because there were not enough of them in the database. For the remaining six species, the database was reviewed, and all photos were downloaded. The photos were sorted out, excluding those that did not contain scenes of damage, thus creating the initial database of raw image data, which was further used for training and testing the classification CNN. The input data contained more than 700 photos of various damages to tree trunks caused by pests. The images were taken at different times of the year, capturing both dead and living trees, different kinds of trees and under various lighting conditions. They were also taken on different devices, so they do not have uniform resolution. These characteristics can be considered positive because they should allow the resulting model to better generalize features present in the input data. However, a potential negative aspect of the used input data is that they were often taken in batches, meaning that the data sometimes include several photos that are very similar to each other. However, this phenomenon cannot be influenced in this work.

Species	No. of raw images
Cerambyx cerdo	96
Ips acuminatus	99
Tomicus minor	88
Ips sexdentatus	141
Pityogenes chalcographus	94
Ips cembrae	75
Ips typographus	208

Tab. 1 – Number available of images for each species

5.2 Classifier design

A common approach in creating image classifiers using convolutional neural networks is the utilization of pre-trained models, which are fine-tuned only on user-selected data classes. Such models typically have a fixed number of input neurons corresponding to the number of pixels in the input image. Training and testing images are often captured at much higher resolutions than the input to pre-trained models. The classical approach is to resize the input images to the desired input dimensions of the model. A simple schematic of the resulting trained classifier operating on this classical principle is shown in Fig. 17, where input I is the resized image to the required size of the model, and output y is the classified class.

Fig. 17 - Scheme of classical design of image classifier

As part of this work, a pilot study was conducted to explore the possibility of utilizing tree damage image data for training a classifier to recognize tree pests using the classical approach to classifier design. In this pilot design, only three classes of pests (*Cerambyx cerdo, Ips acuminatus, Tomicus minor*) were selected, for which a model pre-trained on the GoogLeNet architecture was fine-tuned to investigate the feasibility of this approach.

The resulting model was unable to classify pests based on damage caused to the tree. The model exhibited behaviour where it always classified all tested images as one class, thus minimizing the error. Therefore, the accuracy was 33.3%, as one-third of the testing data always belonged to the class that the model classified all data into. If the model was unable to classify three classes, it made no sense to continue in this direction and attempt to expand it to include additional classes.

After thorough examination of the process and visual inspection of the input and outputs of individual parts, a potential problem was identified. Damage from various pests typically consists of narrow, detailed tunnels chewed into the wood. After a certain amount of comparison of damage from various pests, even a layman can begin to perceive subtle differences in the architecture of the chewed tunnels. The problem arises when the input data (often captured in high resolution) is resized to the input size of 224px x 224px (input size of GoogLeNet architecture). During this enormous spatial reduction, there is an enormous loss of details contained in the original image. Almost no features from the original tunnel structure are preserved, which could be used for training and classifying convolutional neural networks. This approach, referred to as the classical method, thus leads to a dead-end.

Due to the loss of details caused by resizing input data and the subsequent inability to create a classifier, a new approach to classifier design was developed. This new approach, illustrated in Fig. 18, utilizes a pre-processing step of sub-image generation before the classical classifier. The original image I is inputted into this sub-image generator, which then divides it into multiple sub-images $S_1 \ldots S_N$, all of which have the same size corresponding to the input of the model architecture. Each sub-image is then subjected to classification and a corresponding class $C_1 \ldots C_N$ is assigned to it. All these classes corresponding to sub-images then enter the final "Voting" module, where based on the representation of individual classes, a decision is made regarding the classification of the original image as a whole, and a final class y is assigned to it.

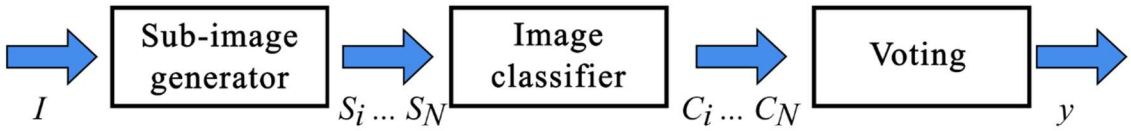

Fig. 18 - Scheme of the new approach in classifier design

Such a division of the input image into small subregions is justified in this particular case. Details about the structure of damage on the tree are primarily stored locally in the image, but they can occur in multiple areas simultaneously, independently of each other. The image as a whole, therefore, does not have as much significance for training and classifying as it might have for other typical classifiers, for example, a classifier of animals, where it is necessary to have as much information as possible about the shape of the animal in the photo. What is important are precisely the small details, occurring abundantly but not necessarily clustered in the training data. Therefore, when input images is divided, we obtain many small images, where many of them contain information about damage details typical for a given pest without loss of detail. At the same time, information about the texture of the wood is preserved on these small, detailed images, which can also help in training and subsequent classification because certain types of bark beetles attack only certain types of trees.

5.2.1 Image classifier

The image classifier is an essential part of the entire design. It accepts image data as input and assigns them to one of the preselected classes based on model training. In this specific case, a convolutional neural network is used as the classifier. For the classifier in this work, a pre-trained CNN architecture, GoogLeNet, was selected [25]. Although it is currently only a concept feasibility study, in the future, this concept could serve as the basis for a mobile real-time application, therefore it is appropriate to choose considering memory and speed requirements. In this case, GoogLeNet appears to be a relatively suitable choice due to its good speed and relatively small size while still maintaining a reasonable level of accuracy.

GoogLeNet is a pre-trained network on the ImageNet database. It has a depth of 22 layers, operates with over 7 million parameters, occupies 27 MB, and works with input coloured images of dimensions 224 px x 224 px [25]. Its interesting feature is its serial-parallel architecture. In addition to the classic serially arranged layers typical for convolutional neural networks, it also includes so-called Inception modules. These modules consist of parallel convolutional layers with kernels of different sizes and one branch with a max-pooling layer. The outputs from these layers are then concatenated into one filter. Within the entire architecture, multiple Inception modules are serially arranged. Thanks to this modification, the network is capable of achieving better accuracies without increasing its depth (which was the classical approach for improving network accuracy), thus saving memory and computational time [26].

5.2.2 Sub-image generator

The task of this module is to pre-process the input image from the original raw image into a group of sub-images with specific dimensions suitable for input into the image classifier architecture, namely 224px x 224px. In this work, two method proposals for the sub-image generator were developed - the Grid division method and the Elliptic division method.

Grid division method

This method operates on an intuitive approach to divide the input image into equal parts using a grid. Its basic principle is illustrated in Fig. 19.

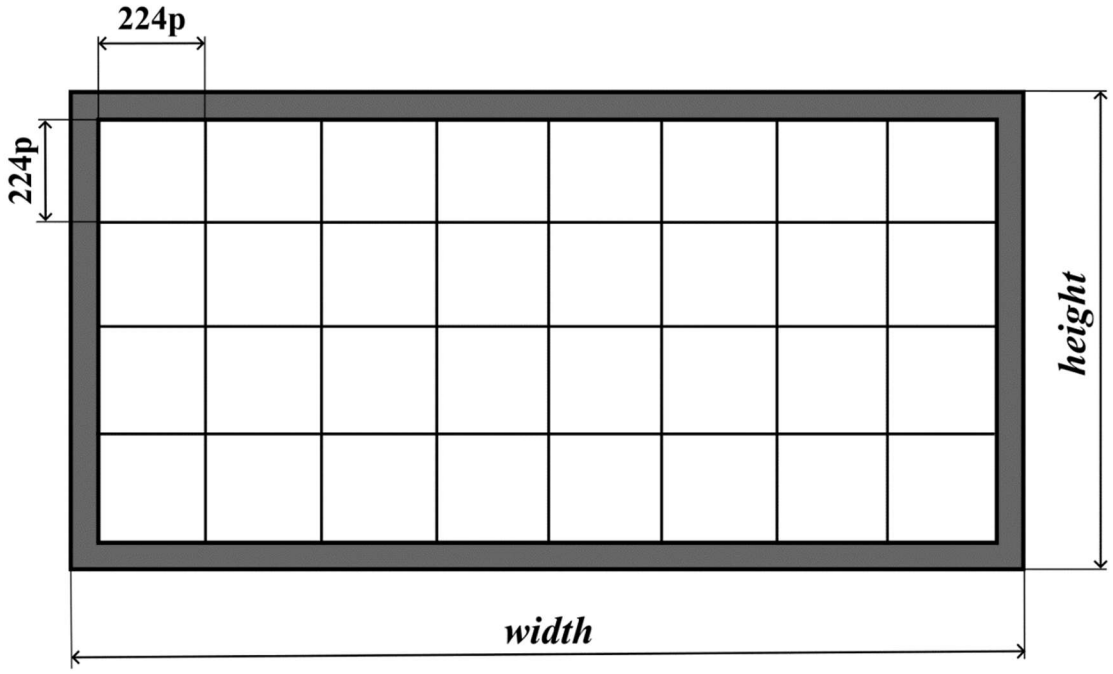

Fig. 19 - Grid division method scheme

First, the dimensions of the input image - height and width - are determined. To divide the image into precisely the same pieces, all with the desired input size for network training (224 px x 224 px), the input image must be cropped (dark grey area on Fig. 19) so that both its height and width are divisible by the input dimension. The number of pixels to be removed was determined using the modulo 224 function, i.e., the remainder after dividing by 224. This information was obtained for the height and width of the image. The first half of these pixels was removed from the top of the image, and the second half from the bottom, or left and right sides, respectively. In the case where the modulo operation resulted in an odd number, one extra pixel was cropped from the bottom or right side. Subsequently, a multitude of smaller images was created from the original image by copying the first square of the grid with dimensions of 224px x 224px and saving it to a folder, followed by taking a crop shifted horizontally or vertically by the width or height of the training image. This process significantly multiplies the number of

images available from one picture, which is a good thing for follow-up training of the model.

It is important to note that many of the newly created input sub-images now had no informative value. Many of them contained scenes or parts of scenes irrelevant to network training, such as sky, leaves, roads, building fragments, and generally parts of unwanted background, especially from the peripheral areas of the original image. The volume of these unwanted images varied for each class, but for some classes, it reached over one-third of all photos, which is a significant amount. To ensure that subsequent classifier training worked well, it was necessary to manually review all these created images and remove all, or at least as many as possible, of the images without informational value. This process was very time-consuming.

Elliptic division method

The previous method, Grid division, is a functional approach for creating and training a convolutional neural network for classifying tree pests based on image data of trunk damage. However, this method has one significant drawback: its requirement for manual cleaning of the input training data after dividing the input image according to the grid. This process is time-consuming. Therefore, the hypothesis arises whether it would be possible to develop a method that could leverage the advantages of the original Grid division method while simultaneously reducing the manual and time-consuming nature of preprocessing training data. This thesis proposes an alternative method that could simplify the preprocessing of training data and create appropriately sized data for input into the training algorithm. This new method is called the Elliptic division method.

The basic idea of this method stems from the consideration that when people commonly take photos of an object with a smartphone or other device, they tend to place the photographed object directly in the center of the frame. Consequently, useful information for training the model is primarily gathered around the center of the image, whereas the edges of the image are unwanted for network training because they contain only background information and not specific tree damage.

The Elliptic division method utilizes the aforementioned principle in the following manner. First, it is detected whether the input image is oriented vertically or horizontally. Then, the center of this image is determined. This center serves as the center of an ellipse that is virtually inscribed into the image. This ellipse has a major axis a and a minor axis b, where the sizes of a and b are obtained according to the following Eq. 3 or Eq. 4 depending on whether the image is oriented vertically or horizontally.

For horizontally oriented images:

$$a = a_r \frac{w}{2}; \qquad b = b_r \frac{h}{2} \tag{3}$$

For vertically oriented images:

$$a = b_r \frac{h}{2}; \qquad b = a_r \frac{w}{2} \tag{4}$$

The parameters a_r and b_r are model meta-parameters determining the ratio of axes to the major dimensions of the image (a_r ratio to original image width w, b_r ratio to original image height h), taking values between (0,1). In this work, both parameters were set to a value of 0.7. Thus, the area of interest ellipse extends to 70% of the height and width of the input image. These parameter values were chosen because the resulting ellipse contains a sufficiently large area with potential informational value while also trimming off a significant portion of the undesired background of the image. After finding the parameters a and b, random coordinates x and y are generated, serving as the center of the newly created sub-image crop. These generated coordinates are inserted into the ellipse equation Eq. 5 or Eq. 6.

For horizontally oriented images:

$$\frac{(x-m)^2}{a^2} + \frac{(y-n)^2}{b^2} < 1 \tag{5}$$

For vertically oriented images:

$$\frac{(x-m)^2}{b^2} + \frac{(y-n)^2}{a^2} < 1 \tag{6}$$

Where m is the x coordinate of center of the original image and n is the y coordinate of center of the original image. All variables are integers because working units are pixels.

The result of this equation must be less than 1, indicating that the generated center coordinates of the sub-image lie within the virtual ellipse. Otherwise, the coordinates are discarded, and new ones are generated. Once the center of the sub-image lying inside the ellipse is found, a crop of appropriate dimensions 224px x 224px is created around this center, which is then saved to a folder and will be used as input for training the model in the later stage. This process is repeated N times, where N is the number of sub-images created from one input photo. The principle of the Elliptic division method is schematically illustrated in Fig. 20.

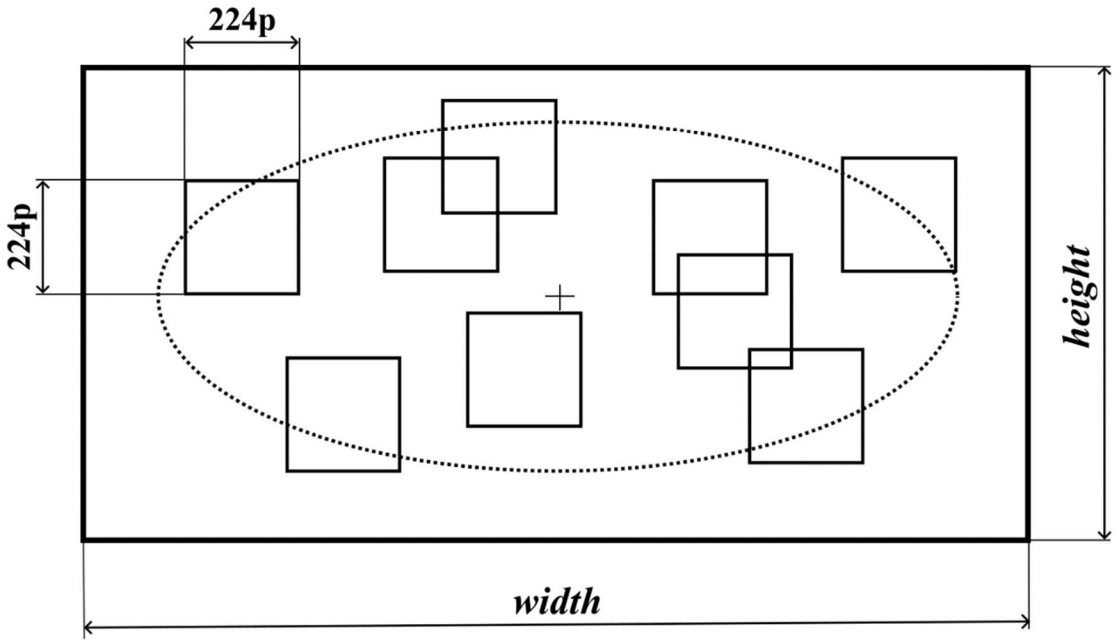

Fig. 20 - Elliptic division method scheme

After applying the Elliptic division method, the number of unwanted sub-images decreased significantly compared to the Grid division method.

5.2.3 Voting system

The output from the image classifier is a set of sub-images, with each assigned to one of the classification classes. To decide on the classification of the input image as a whole, it's necessary to integrate information from individual sub-images, and that's where the voting module comes into play. First, the frequency of occurrence of each class f_j is determined. The final classification is then determined by the relationship

$$y = argmax\left(f_j\right) \tag{7}$$

where f_j is the frequency of occurrence of the class j in the set of sub-images. The class y for the input image X is thus selected as the most frequently represented class among all sub-images.

5.3 Datasets

From the provided raw data, two datasets were created - a training dataset and a testing dataset. Since all classes have different data distributions, the first step was to limit all classes to the same number of images based on the least represented class. Therefore, the working data included 75 photos from each class to achieve a uniform balance. The selection of these photos was random for classes with a higher number of images. Next, the working data was divided into training and testing sets. The data was split in a ratio of 85:15, meaning 64 photos were used for training the model and 11 photos were used

for testing. Due to the low number of input data, a validation dataset was not created. The training and testing datasets thus created were used for all methods and their variations examined in this work (pilot-classical method, Grid division method, Elliptic division method). Random selection of input data and subsequent division into training and testing sets was performed only once. Therefore, all case studies work with the same training and testing data for better result comparison.

5.3.1 Training set

Grid division method training set

The training dataset for the classifier model was created using pre-processing through the sub-image generator using the Grid division method. This process generated several thousand sub-images for each class. It's important to note that many of these generated data had no informational value for model training. They included various parts of the background, sky, leaves, paths, buildings, etc. Such unwanted sub-images constituted up to a third of the total data for some classes. All data for each class were manually reviewed, and all or at least most of the irrelevant data were removed. This manual cleaning of training data was a very time-consuming process. As a result, there was a need to develop a new method for dividing input data into sub-images with lower manual preprocessing requirements, namely the elliptic method.

Tab. 2 shows an overview of the number of sub-images for each class after manual cleaning. The final training dataset was created with respect to the least represented class to ensure all classes were evenly represented. The number of data for a class in the final dataset corresponds to the number of data in the least represented class, while data for classes with a larger number of data were randomly selected.

Species	No. of preprocessed images
Cerambyx cerdo	3194
Ips acuminatus	2634
Tomicus minor	2923
Ips sexdentatus	5058
Pityogenes chalcographus	3418
Ips cembrae	2648
Ips typographus	5145

Tab. 2 - Overview of No. of cleaned data

Elliptic division method training set

Similarly to the previous case, the training data was generated in the sub-image generator, this time using the Elliptic division method. This method was designed to significantly reduce the generated number of irrelevant sub-images by its nature. The Elliptic division method allows generating any number of sub-images from a single input image. In this

case, the generated number was set to $N = 40$, as the resulting number of training data (2560 sub-images per class) roughly matched the number of training data when using the Grid division method, enabling better comparison of these methods. Theoretically, an increase in the number of generated images could enhance model training due to a larger amount of training data. However, excessive generation of images might result in them becoming very similar to each other, potentially leading to undesired overfitting during training.

The numbers of training data for individual methods are summarized in Tab. 3.

Method	No. of training images
Grid division	2634
Elliptic division	2560

Tab. 3 - No. of training images for each method

5.3.2 Training

To train the image classifier, the principle of transfer learning was utilized. Initially, a chosen pre-trained architecture of the convolutional neural network, GoogLeNet, was employed. From this network, the last three layers were removed and replaced with new layers of the same type. Specifically, this involved a fully connected layer, which connects the outputs of the last convolutional layer, a softmax layer, which recalculates the resulting values into probabilities of belonging to individual classes, and a classification layer responsible for classifying into the newly chosen classes of the model. The modified model did not need to be trained from scratch but only fine-tuned for the purpose of identifying tree pests, which reduced the time complexity of the training process.

For further increase in input data and to mitigate overfitting effects during training, image augmentation is applied, allowing random mirroring of input images along the x and y axes, resizing images by +- 20%, and rotation of images in the range of 0-360°.

After reviewing the official MATLAB forums, the ADAM (Adaptive Moment Estimation) algorithm was chosen as the optimizer, which seemed to be a good choice for transfer learning and should have less sensitivity to other selected training parameters [27]. An overview of all training parameter settings is shown in Tab. 4.

Optimizer	ADAM
Learn rate schedule	Piecewise
Learn rate drop factor	0.1
Learn rate drop period	10 epochs
Maximum epochs	50 epochs
Mini batch size	50 images
Shuffle	Every epoch
Initial learn rate	0.0001
Gradient decay factor	0.9
Squared gradient decay factor	0.999
Epsilon – Denominator offset	$1 \cdot 10^{-8}$

Tab. 4 - Training parameters, Grid division method

The classification layer of the model computes the loss function as the cross-entropy loss described by

$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} w_i t_{ni} \ln y_{ni} \qquad (8)$$

where N is the number of samples, K is the number of classes, w_i is the weight for class i, t_{ni} is the indicator that sample n belongs to class i, and y_{ni} is the output for sample n for class i, which is in this particular case value from softmax layer. Stated differently, y_{ni} represents the propability that the network links observation n to class i [28]. This settings were used for all methods and cases.

5.4 Case study

After it was found in the pilot study that the conventional approach to classifier design would not be applicable for classifying pests based on image data of damage, a case study approach was chosen to investigate whether the newly proposed methods were capable of classification and to examine the behavior of the model with higher numbers of classification classes.

Initially, models were trained to recognize 3 classes of pests, specifically *Cerambyx cerdo, Ips acuminatus* and *Tomicus minor*. These species were selected because damage to the tree trunk by these species appeared to be most distinct to the layman's eye, thus theoretically making it easiest to train the model to recognize these classes. The main objective of this case was to determine whether it was possible to train the model to recognize tree damage.

In the second case, the possibility of classifying 5 species was tested. The additional species were *Ips sexdentatus* and *Pityogenes chalcographus*, chosen randomly. In the third case, the model's ability to be trained for classification of all 7 selected types of bark

beetles was tested. The aim of the studies with 5 and 7 species was to determine how well the trained model performed with an increasing number of more similar classes.

5.5 Evaluation

As the main metric for evaluating and comparing the trained models for individual methods and classes, accuracy was utilized. Accuracy is one of the fundamental metrics for describing classification models. This metric provides information about the ratio of correctly classified cases to all cases and can be described by

$$\text{Accuracy} = \frac{CCS}{AS} \tag{9}$$

where CCS stands for correctly classified samples and AS for number of all samples.

For a deeper insight into the classification abilities of the model, the second metric chosen was the Confusion Matrix. A confusion matrix is a matrix with the model's predicted classes on the horizontal axis and the actual classes on the vertical axis. Thus, on the main diagonal, we get the number of correctly classified samples into their respective classes. Entries outside the main diagonal provide information about how many samples from which class were misclassified into another class and into which one. With this information, we can observe, for example, whether the model tends to systematically misclassify a particular class as another specific class or whether the classification of a certain class is generally more problematic than the classification of other classes.

5.6 Implementation

The entire practical part of this thesis was developed in the MATLAB 2023a programming environment. Within this environment, the Deep Learning Toolbox extension was utilized, allowing for expanded capabilities in working with neural networks. The main part of the work is divided into six scripts, which are attached to this thesis. Essentially, there is a script dedicated to the preprocessing phase (*_preprocessing.m*), a script focused on setting up and training models (*_training.m*), and a testing script where the results are evaluated (*_testing.m*). These three scripts were created for both the Grid division and Elliptic division methods (corresponding files have the respective prefixes *grid_* and *elliptic_*).

6 RESULTS

The following Tab. 5 shows accuracy results for both methods – Grid division and Elliptic division.

	3 classes	5 classes	7 classes
Grid division method	**96,97 %**	**90,91 %**	**85,71 %**
Elliptic division method	**96,97 %**	**90,91 %**	**88,31 %**

Tab. 5 - Comparison of accuracies of trained models and methods

On **Chyba! Nenalezen zdroj odkazů.** – Fig. 23 are confusion metrices for cases with 3, 5 and 7 classes, respectively, for Grid division method.

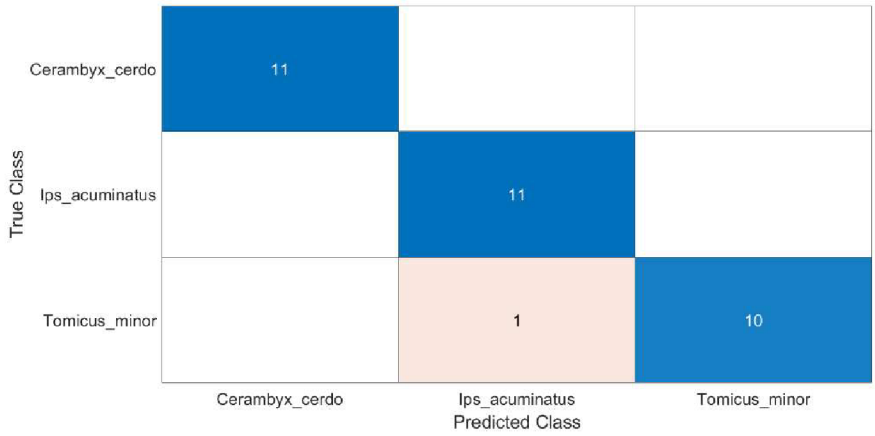

Fig. 21 - Grid division method, confusion matrix - 3 classes

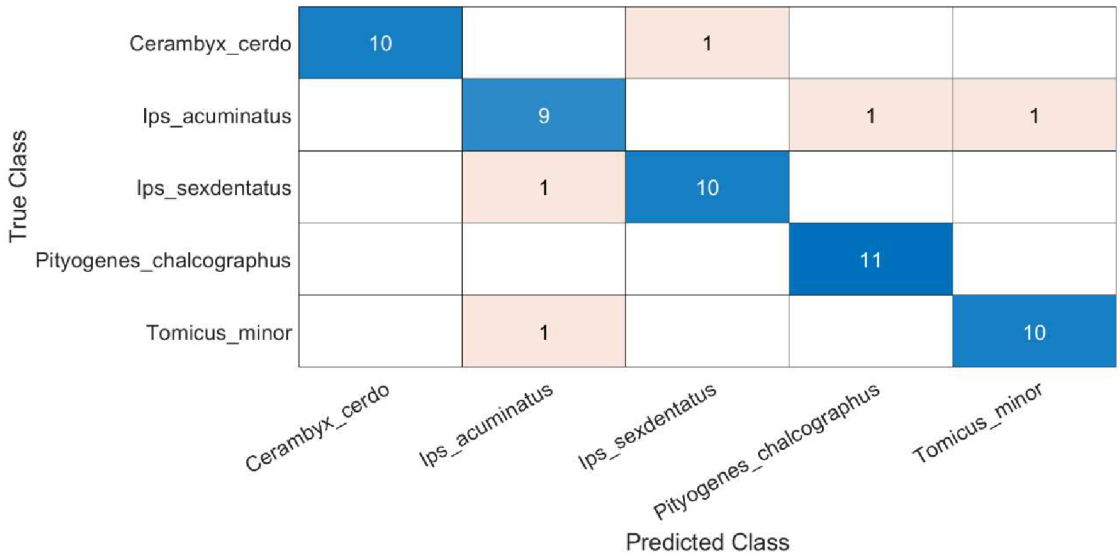

Fig. 22 - Grid division method, confusion matrix - 5 classes

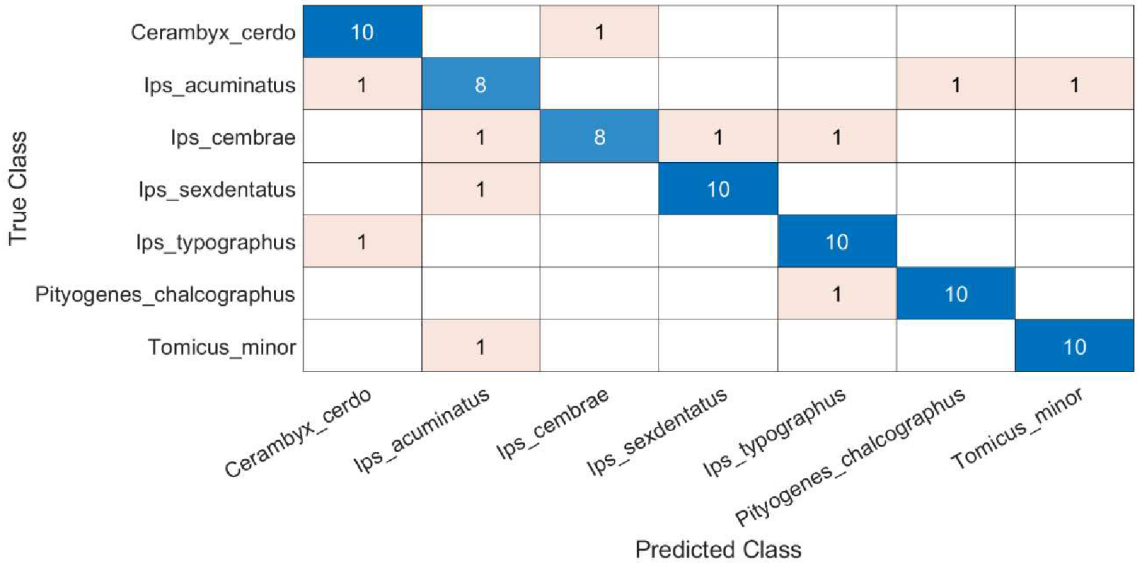

Fig. 23 - Grid division method, confusion matrix - 7 classes

On Fig. 24 – Fig. 26 are confusion metrices for cases with 3, 5 and 7 classes, respectively, for Elliptic division method.

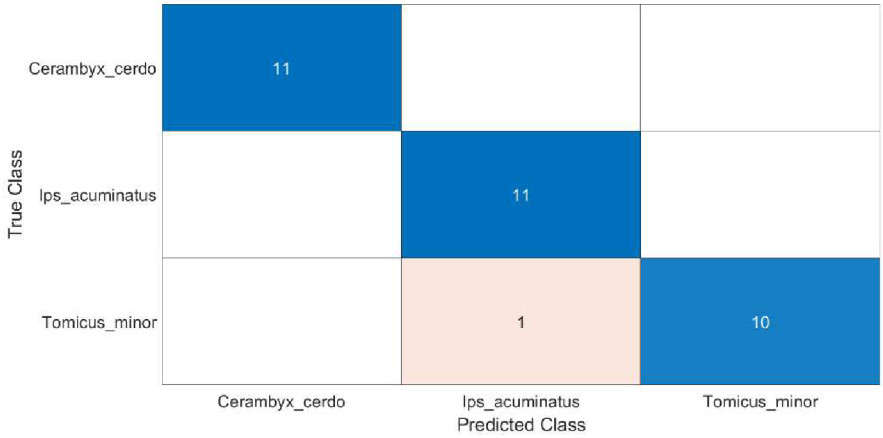

Fig. 24 - Elliptic division method, confusion matrix - 3 classes

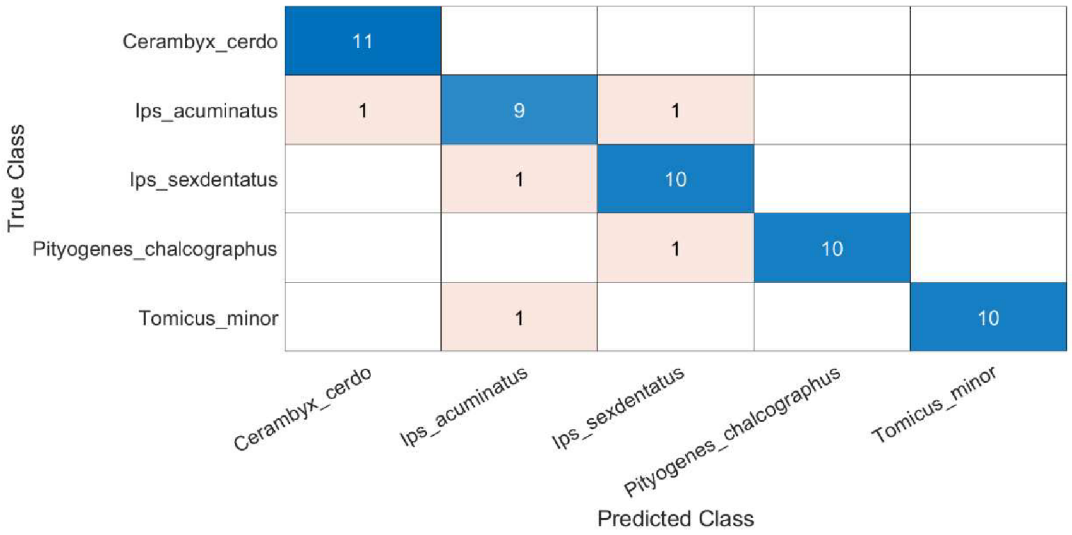

Fig. 25 - Elliptic division method, confusion matrix - 5 classes

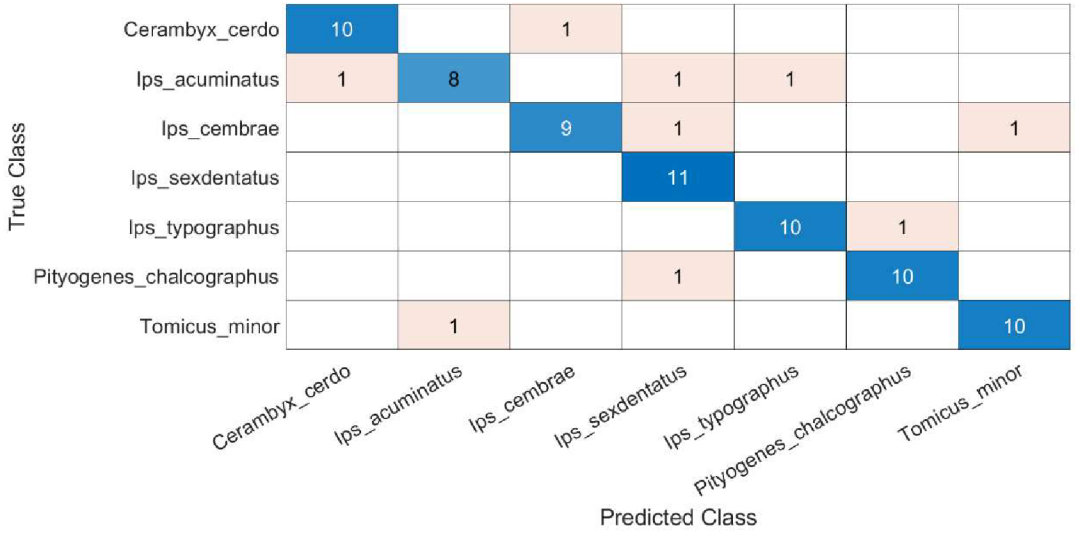

Fig. 26 - Elliptic division method, confusion matrix - 7 classes

7 DISCUSION

As part of the feasibility assessment of using image data of tree damage for pest classification, initial research was conducted using convolutional neural networks for the task. Commonly used approaches for creating classifiers utilize CNNs by resizing the original image data intended for network training to the required dimensions usable for fine-tuning the pretrained network. This procedure, referred to as the classical method in this work, was tested as the first pilot variant for the pest classifier. However, the results of the classical method were extremely poor. Only for three presented species of pests, the trained model was unable to classify. It consistently classified all tested images as a single class, attempting to minimize incorrect predictions. When all images were classified into the same class, the accuracy reached 33.33%, meaning it was correct in one third of cases since the testing data had an equal number of images for each class (11 images per class). From this behaviour of the model, it became evident that continuing this approach in creating a pest classifier based on image data of tree damage was futile.

The negative behaviour of the classical method can be explained by the fact that resizing the input data leads to excessive degradation of image details. The input data of tree trunk damage typically contain complex tunnels and holes in the wood caused by pests. Each pest species forms these tunnels and holes differently, with varying proportions of tunnel width and length, different orientations of tunnels relative to each other, or different patterns of tunnel creation. All these elements are highly detailed and often have similar colors to the surroundings. When resized to the required size for input into the training process, these details disappear, making them unrecognizable to the model.

As a solution to this problem, a sub-image generator was proposed to precede the classifier. Two methods for the sub-image generator were proposed - the Grid division method and the Elliptic division method. Tab. 5 compares the resulting accuracies of the trained models using these methods for three case studies. From the accuracy results, it can be seen that both proposed methods - Grid division and Elliptic division - achieve mutually comparable results. Moreover, these results can be considered very good, especially considering the limited training data available. In comparison of these two methods, the second one created, the Elliptic division method, emerges as more suitable.

It utilizes the same principle of dividing the original image into smaller sub-images suitable in size for training a convolutional neural network model as a Grid division method. However, it significantly reduces the number of sub-images that do not contain any useful data for training, thereby reducing the time required for preprocessing the input data, as it is not necessary to go through all created sub-images and remove the unwanted ones manually. Another potential advantage of the Elliptic division method over the Grid division method is that it allows users to choose how many sub-images are created from one input image. Thanks to this feature, it is easy and quick to increase the training data

set, leading to the training of a more accurate model. However, it is necessary to note that with an increasing number of created sub-images, these sub-images will become more and more similar to each other, which may ultimately have a negative impact on the model and may lead to model overfitting. Therefore, it is still preferable to have as large a database of input training images as possible to enable the convolutional neural network model to capture all the important features for each classified class as accurately as possible.

Upon closer examination of the confusion matrices, two most problematic classes can be identified for both methods: *Ips acuminatus* and *Ips cembrae*, which the model struggles the most to classify. Nevertheless, the model still correctly classifies the majority of these test data (8 out of 11 for each of the problematic classes for Grid division method and 8 out of 11 and 9 out of 11, respectively, for Elliptic division method). Higher number of training images could increase model's ability to correctly classify all classes.

Several suggestions could lead to improving the created pest classifier in the future. Firstly, it would be appropriate to test the impact of the resulting model's accuracy on the number of selected created sub-images from one training image using the Elliptic division method. What is the optimal number? When, if at all, does overfitting occur? Furthermore, it would be suitable to apply the created model to a completely different set of test data to verify its functionality. To overall improve the model's behaviour, having higher quality input data specifically collected for this purpose would certainly help. Data collected from different times of the year, different types of trees, and different trees of the same species could all contribute to improving the accuracy of the model. This could also be combined with adding new pest species for classification based on discussions with experts, which would have significance for the intended use of such a classifier. Overall, these initial results can be summarized by stating that it is possible to create a classifier for tree pests based on image data of tree damage, assuming we work with sub-images of original data that retain information about damage details.

8 CONCLUSION

The possibilities of utilizing image data of tree damage caused by pests for the classification of these pests have been explored in this study. The study focuses on investigating the possibility of creating a classifier based on convolutional neural networks. The diploma thesis was created in collaboration with National Forest Centre, Forest Research Institute, Slovakia, which provided a database of labelled image data of tree damage, serving as input data for training the classifier.

The data used to create the classifier, originating from database [24], are from real-world environments and thus vary in lighting conditions, shadows, photo dimensions, etc. The data come from various time periods throughout the year. They include tree damage caused by various pests on different tree species, whether living or dead trees. As a result, the created classifier can effectively capture various real-life conditions. The original database contained hundreds of types of damage caused by pests, as well as diseases, fungi, or animals. The database was iteratively manually reviewed, and records that did not fit the essence of this work were removed. Initially, all records except those of pest-caused damage were removed. Subsequently, the species were narrowed down to those containing the highest number of photos. From these, the type of damage for which the classifier was designed was selected - damage to the trunk of the tree. After discussion with experts, the selection was narrowed down to the last seven species corresponding to the types of bark beetles commonly found in Czech and Slovak forests, for which the classifier was designed: *Cerambyx cerdo, Ips acuminatus, Ips cembrae, Ips sexdentatus, Ips typographus, Pityogenes chalcographus,* and *Tomicus minor*. The data were divided into training and testing datasets in a ratio of 85:15. To ensure an equal number of data points for each class, all counts were standardized based on the least represented class. For training, 64 images were used, and for testing, 11 images from each class.

The practical part of the work focuses on designing a image classifier for tree pests utilizing convolutional neural networks. Due to the low number of training data, the approach of utilizing transfer learning and fine-tuning an existing model to classify pest classes was chosen. The pre-trained network chosen was GoogLeNet, which generally achieves good accuracy and speed with small memory requirements, suitable for potential future use of such a classifier in a mobile application.

At first, a pilot study was conducted with three classes of pests. In this study, a classical approach was employed, where the input data was resized to the size required by the pre-trained model (224px x 224px). Model trained this way was unaible to classify.

Upon inspecting the entire process, it was concluded that the model cannot classify due to significant loss of image details during resizing of input data. Damage to trees by pests is inherently a detailed matter, and with the loss of these details, the original structures of damage in the image cannot be distinguished. Therefore, the use of the classical method cannot be applied to create a classifier.

As a solution to this problem, a sub-image generator was proposed to precede the classifier. This generator divides the input image into a set of sub-images with the desired dimensions without reducing the quality of details. As a sub-image generator, two methods were designed. The Grid division method was proposed, which divides the input image into equal squares of the size corresponding to the input data for training the model (224px x 224px) according to the grid. This preserves the details, and additionally, it increases the number of training data. By cutting the input image, a series of sub-images is created, which can be used for training the model, but it also creates a considerable amount of unwanted sub-images containing background parts – sky, leaves, healthy parts of the tree, parts of the forest, buildings, etc. Such data would negatively affect the model training and thus had to be removed. For some classes, the unwanted data comprised up to a third of all training data. All training data had to be manually reviewed class by class, and all of unwanted images, or at least as many as possible of them, had to be manually removed. This process was necessary but very time-consuming. However, the resulting trained model, unlike the classical method, was able to classify the input data very well according to the pest that caused the damage. The model, initially trained for three species, was later expanded to five and seven species, where even with the increasing number of classes, it still classified the input data very accurately.

Given the significant time-consuming nature of the Grid division method during the preprocessing of training data, a new method - the Elliptic division method - was developed. This method operates on the idea that input data come from a real-world environment rather than a laboratory setting. When people typically take photos, they tend to place the object of interest in the center of the frame. From this notion, it follows that all relevant information about tree damage is preserved around the center of the image. Therefore, the Elliptic division method creates sub-images from the original image only within a virtual ellipse inscribed inside the original image, whose major and minor axes are reduced by a certain ratio, a_r and b_r, concerning the original width and height of the image. The resulting sub-images thus utilize the advantage of detail preservation observed in the previous method but also significantly reduce the proportion of unwanted sub-images, which would negatively affect the model training. A model trained on these created sub-images achieved very good ability to classify input data based on image data of tree damage.

From this work, it follows that tree trunk damage image data can be used for pest classification. The Elliptic division method is favoured over other Grid division method due to its lower data preprocessing requirements and higher flexibility in generating sub-images of training data. These findings open up possibilities for using these methods for the future development of a real-time application, expanded to include additional types of pests, which could be used by forestry workers to facilitate pest monitoring and prevention of their outbreaks.

9 REFERENCES

[1] ELLISON, David, Cindy E. MORRIS, Bruno LOCATELLI, et al. Trees, forests and water: Cool insights for a hot world. *Global environmental change* [online]. OXFORD: Elsevier, 2017, **43**, 51-61 [cit. 2024-04-04]. ISSN 0959-3780. Available from: doi:10.1016/j.gloenvcha.2017.01.002

[2] PEARLMUTTER, David, Carlo CALFAPIETRA, Roeland SAMSON, Liz O'BRIEN, Silvija KRAJTER OSTOIĆ, Giovanni SANESI a Rocío ALONSO DEL AMO. *The Urban Forest: Cultivating Green Infrastructure for People and the Environment*. 7. Cham: Springer International Publishing, 2017, 362 s. ISBN 3319502794. ISSN 1876-0899. Available from: doi:10.1007/978-3-319-50280-9

[3] DALE, Virginia H, Linda A JOYCE, Steve MCNULTY, et al. Climate Change and Forest Disturbances. *Bioscience* [online]. Circulation, AIBS, 1313 Dolley Madison Blvd., Suite 402, McLean, VA 22101. USA: American Institute of Biological Sciences, 2001, **51**(9), 723-734 [cit. 2024-04-04]. ISSN 0006-3568. Available from: doi:10.1641/0006-3568(2001)051[0723:CCAFD]2.0.CO;2

[4] PIMENTEL, David, S. MCNAIR, J. JANECKA, et al. Economic and environmental threats of alien plant, animal, and microbe invasions. *Agriculture, ecosystems & environment* [online]. AMSTERDAM: Elsevier B.V, 2001, **84**(1), 1-20 [cit. 2024-04-04]. ISSN 0167-8809. Available from: doi:10.1016/S0167-8809(00)00178-X

[5] MARINI, Lorenzo, Matthew P. AYRES a Hervé JACTEL. Impact of Stand and Landscape Management on Forest Pest Damage. *Annual Review of Entomology* [online]. 2022, (67), 181-199 [cit. 2024-04-04]. ISSN 1545-4487. Available from: doi:10.1146/annurev-ento-062321-065511

[6] LIU, Corsa Lok Ching, Oleksandra KUCHMA a Konstantin V. KRUTOVSKY. Mixed-species versus monocultures in plantation forestry: Development, benefits, ecosystem services and perspectives for the future. *Global ecology and conservation* [online]. AMSTERDAM: Elsevier, 2018, **15**, e00419 [cit. 2024-03-24]. ISSN 2351-9894. Available from: doi:10.1016/j.gecco.2018.e00419

[7] BARBEDO, Jayme Garcia Arnal. Detecting and Classifying Pests in Crops Using Proximal Images and Machine Learning: A Review. *AI (Basel)* [online]. Multidisciplinary Digital Publishing Institute, 2020, **1**(2), 312-328 [cit. 2024-03-24]. ISSN 2673-2688. Available from: doi:10.3390/ai1020021

[8] EBRAHIMI, M. A., M. H. KHOSHTAGHAZA, S. MINAEI a B. JAMSHIDI. Vision-based pest detection based on SVM classification method. *Computers and electronics in agriculture* [online]. OXFORD: Elsevier, 2017, **137**, 52-58 [cit. 2024-03-24]. ISSN 0168-1699. Available from: doi:10.1016/j.compag.2017.03.016

[9] YAO, Qing, Jun LV, Qing-jie LIU, Guang-qiang DIAO, Bao-jun YANG, Hong-ming CHEN a Jian TANG. An Insect Imaging System to Automate Rice Light-Trap Pest

Identification. *Journal of Integrative Agriculture* [online]. OXFORD: Elsevier B.V, 2012, **11**(6), 978-985 [cit. 2024-03-24]. ISSN 2095-3119. Available from: doi:10.1016/S2095-3119(12)60089-6

[10] LI, Wenyong, Tengfei ZHENG, Zhankui YANG, Ming LI, Chuanheng SUN a Xinting YANG. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. *Ecological informatics* [online]. Elsevier B.V, 2021, **66**, 101460 [cit. 2024-03-24]. ISSN 1574-9541. Available from: doi:10.1016/j.ecoinf.2021.101460

[11] SUN, Yu, Xuanxin LIU, Mingshuai YUAN, Lili REN, Jianxin WANG a Zhibo CHEN. Automatic in-trap pest detection using deep learning for pheromone-based Dendroctonus valens monitoring. *Biosystems engineering* [online]. SAN DIEGO: Elsevier, 2018, **176**, 140-150 [cit. 2024-03-25]. ISSN 1537-5110. Available from: doi:10.1016/j.biosystemseng.2018.10.012

[12] CHO, J., J. CHOI, M. QIAO, C. W. JI, H. Y. KIM, K. B. UHM a T. S. CHON. Automatic identification of whiteflies, aphids and thrips in greenhouse based on image analysis. *International Journal of Mathematics and Computers in Simulation* [online]. 2007, (1), 46-53 [cit. 2024-03-25]. Available from: https://api.semanticscholar.org/CorpusID:51290399

[13] ZHONG, Yuanhong, Junyuan GAO, Qilun LEI a Yao ZHOU. A vision-based counting and recognition system for flying insects in intelligent agriculture. *Sensors (Basel, Switzerland)* [online]. BASEL: Mdpi, 2018, **18**(5), 1489 [cit. 2024-03-25]. ISSN 1424-8220. Available from: doi:10.3390/s18051489

[14] GUO, Ying, Junjia GAO, Xuefeng WANG, et al. Precious Tree Pest Identification with Improved Instance Segmentation Model in Real Complex Natural Environments. *Forests* [online]. Basel: MDPI, 2022, **13**(12), 2048 [cit. 2024-03-25]. ISSN 1999-4907. Available from: doi:10.3390/f13122048

[15] BURKOV, Andriy. *The hundred-page machine learning book*. Quebec, Canada: Andriy Burkov, 2019, xviii, 141 stran : barevné ilustrace. ISBN 978-1-9995795-0-0.

[16] SINDHU, Velu, S. NIVEDHA a M. PRAKASH. An empirical science research on bioinformatics in machine learning. *Journal of mechanics of continua and mathematical sciences* [online]. 2020, (Sp. 7) [cit. 2024-03-24]. Available from: doi:10.26782/jmcms.spl.7/2020.02.00006

[17] JAMES, Gareth (Gareth Michael), Daniela WITTEN, Trevor HASTIE a Robert TIBSHIRANI. *An introduction to statistical learning: with applications in R*. Second edition. New York: Springer, 2021, xv, 607 stran : ilustrace ; 24 cm. ISBN 978-1-0716-1417-4.

[18] WITTEN, Ian H, Eibe FRANK a Mark A HALL. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd ed. San Diego: Elsevier Science, 2011, 664 pages. ISBN 9780080890364.

[19] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep learning*. MIT Press, 2016.

[20] CAPUTO, Rodrigo Ribeiro, Edimilson Batista DOS SANTOS a Leonardo Chaves Dutra DA ROCHA. Convolutional neural networks applied to data organized as OLAP cubes. *Expert systems* [online]. Oxford: Blackwell Publishing, 2023, **40**(10) [cit. 2024-03-24]. ISSN 0266-4720. Available from: doi:10.1111/exsy.13454

[21] SAHA, Sumit. Comprehensive Guide to Convolutional Neural Networks. *Towards Data Science* [online]. 2018 [cit. 2024-03-24]. Available from: https://towardsdatascience.com/a-comprehensiveguide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[22] YINGGE, Huo, Imran ALI a Kang-yoon LEE. Deep Neural Networks on Chip - A Survey. In: *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)* [online]. IEEE, 2020, s. 589-592 [cit. 2024-03-24]. Available from: doi:10.1109/BigComp48618.2020.00016

[23] THE MATHWORKS, INC. Pretrained Deep Neural Networks. *MathWorks* [online]. 2024, 1994-2024 [cit. 2024-03-25]. Available from: https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

[24] LESNÍCKA OCHRANÁRSKA SLUŽBA. *Atlas škodcov lesa* [online]. 2024 [cit. 2024-03-25]. Available from: https://www.skodcoviadrevin.sk/

[25] THE MATHWORKS, INC. GoogLeNet. THE MATHWORKS, INC. *MathWorks* [online]. 2024, 1994-2024 [cit. 2024-03-25]. Available from: https://www.mathworks.com/help/deeplearning/ref/googlenet.html?s_tid=doc_ta

[26] SZEGEDY, Christian, Wei LIU, Yangqing JIA, et al. Going Deeper with Convolutions. In: *ArXiv.org* [online]. Ithaca: Cornell University Library, arXiv.org, 2015 [cit. 2024-03-25]. Available from: doi:10.48550/arxiv.1409.4842

[27] THE MATHWORKS, INC. Options for training deep learning neural networks. THE MATHWORKS, INC. *MathWorks* [online]. 2024, 1994-2024 [cit. 2024-03-25]. Available from: https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html

[28] THE MATHWORKS, INC. Classification Layer. THE MATHWORKS, INC. *MathWorks* [online]. 2024, 1994-2024 [cit. 2024-04-25]. Available from: https://www.mathworks.com/help/deeplearning/ref/classificationlayer.html?s_tid=doc_ta

LIST OF IMAGES

LIST OF TABLES

LIST OF APPENDICES

MATLAB scripts:

grid_preprocessing.m
grid_training.m
grid_testing.m
elliptic_preprocessing.m
elliptic_training.m
elliptic_testing.m