TECHNISCHE UNIVERSITÄT DRESDEN
FAKULTÄT INFORMATIK

Master Thesis

for obtaining the academic degree Master of Science

An Approach to Self-Supervised Object Localisation through Deep Learning Based Classification

Andrei Politov

February 25, 2021

1st Reviewer: Dr. rer. medic. Nico Scherf
2nd Reviewer: Dr. rer. nat. Peter Steinbach

Dresden

Declaration of originality

I hereby declare that I have submitted my master's thesis to the examination board of the Faculty of Computer Science on the subject of:

An Approach to Self-Supervised Object Localisation through Deep Learning based classification

I wrote it completely independently and did not use any sources or aids other than those specified, and I marked citations.

Dresden, February 25, 2021
Andrei Politov

Abstract

Deep learning has become ubiquitous in science and industry for classifying images or identifying patterns in data. The most widely used approach to training convolutional neural networks is supervised learning, which requires a large set of annotated data. To elude the high cost of collecting and annotating datasets, self-supervised learning methods represent a promising way to learn the common functions of images and videos from large-scale unlabeled data without using human-annotated labels. This thesis provides the results of using self-supervised learning and explainable AI to localise objects in images from electron microscopes. The work used a synthetic geometric dataset and a synthetic pollen dataset. The classification was used as a pretext task. Different methods of explainable AI were applied: Grad-CAM and backpropagation-based approaches showed the lack of prospects; at the same time, the Extremal Perturbation function has shown efficiency. As a result of the downstream localisation task, the objects of interest were detected with competitive accuracy for one-class images. The advantages and limitations of the approach have been analysed. Directions for further work are proposed.

Contents

1 Introduction

Machine learning technologies are a fast-growing area of artificial intelligence (AI). These technologies have been used in many promising industry projects and scientific research, which has allowed machine learning to develop widely in recent years [93], [40], [16], [82].

Over the past decade, there have been a number of remarkable achievements in machine learning and, in particular, in deep learning techniques based on artificial neural networks [23].

A long history of AI research started in 1956 continuing nowadays in the epoch of "third revolution of neural networks" as some researchers named it [106] [79]. Increasing the number of publications [120], [128] and in general, the progress in the field of neural networks observed since the end of the 2000s. One of the key factors was the development and improvement of quality and size of datasets [9]. Also, computing power that can handle them has appeared. It stands to mention, that the widespread adoption of GPUs (and later appearance of professional GPUs), virtually unlimited storage options, and the development of big data technologies have become a necessary basis for the evolution of deep learning [23].

The presence of large datasets gave an opportunity for the development and adoption of deep learning. However, creating large-scale datasets requires collecting and annotating a vast amount of data. This process is time-consuming and expensive. One of the possible solutions is learning visual features from large-scale unlabeled data (images or videos) without using any human annotations. In other words, formulate an unsupervised learning problem as a supervised one. Replacing the human annotation by creatively using some properties of the data to create a pseudo supervised task. This is the self-supervised learning approach.

Self-supervised learning has achieved impressive results in last years [42], [59], [84]. The method is a promising technique and functional tool for scientists in different areas [53] [121]. Self-supervised learning is becoming one a promising approach in areas such as Nanoscience, Material science, Biology [125], [19]. Scanning electron microscopy (SEM) is one of the main tools for researchers from these branches. SEM can produce a large number of images. Humans for labelling the data after every experiment are impossible to come by.

The ability to identify, recognise or localise a specific type of object for extensive datasets produce a particular interest in Self-Supervised Learning from scientists. [53], [19], [50].

Within this thesis, the theory and practice of using self-supervised learning as part of a deep learning task are studied. A classification task, object recognition and localisation problem are presented in details. An overview of the methods of explainable AI and their classification is given. The use of a classification task with a subsequent application of methods of explainable AI, and finally, the use of a localisation task for the detection of objects on different SEM-based datasets will be shown. In the final part of the thesis, conclusions about the work are presented. The results are summarized and directions for further work are proposed.

2 Problem Statement and Datasets

The scientific task for the work will be presented in the chapter. Moreover, the datasets used in the thesis will be presented in detail in the following chapter. Three datasets were used. The concept and the structure of the data in all datasets are similar and will be described below.

2.1 Problem Statement

The main goal of this work is to use Deep Neural Networks and a combination of Self-Supervised learning and Explainable AI to localize objects in images obtained with a scanning electron microscope (SEM).

This results in the following sub-tasks:

1. Use a classification task as a pretext task of self-supervised learning task to classify signal/non-signal images in different datasets;
2. Use methods of explainable AI to extract representations that are sensitive to the matter responsible for the network's output;
3. Apply localisation methods by using explainable AI representations to count the objects in the signal images of the datasets.

2.2 Datasets

2.2.1 Diamond dataset

The diamond dataset was created from the images of a scanning electron microscope. The object of observation was the results of the laser beam experiment based on [31]. In this experiment, a metal disc was placed inside an accelerated plasma. After bombardment, the disk was placed under a Scanning Electron Microscope and several regions of the disc were imaged.

According to the authors of the dataset, the experiment progress was: "The initial motivation was heating a polystyrene foil which is backed by a metal plate. The laser ablates the material and creates a shock wave that traverses the foil. Upon exit from the foil, the diamonds exit the foil as well and are deposited on the metal plate. The metal plate is then analyzed with a scanning electron microscope".

The dataset contains 1553 (1280*960 greyscale) images from the electron microscope in total. The dataset contains images with different level of magnification from m01 till m31. The most interesting part of the dataset is the m03-m06 magnification images. This part counts 387 images.

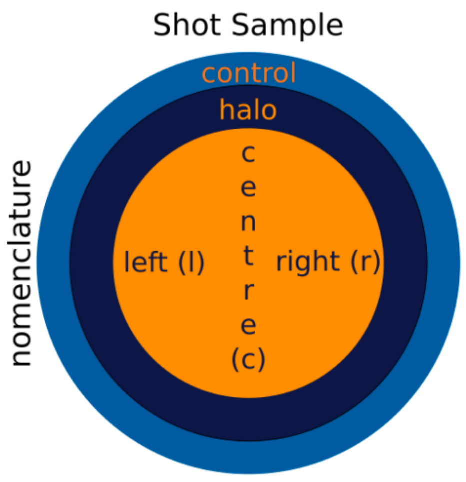

Figure 1: Diagram of the spatial distribution of instances on the metal disc on a diamond dataset [116]

Images from the dataset categorized according to the figure 2:

- centre (‘c‘)
- right (‘r‘) from the centre
- left (‘l‘) from the centre
- ‘halo‘ or ‘halo2‘
- ‘control‘ (on the edge)

The core has the highest density of diamonds images. The second-highest density is in the halo region. No diamonds or just an insufficient amount is in the control region.

This is a fully blind sample. Named contamination control sample (CCS). A metal disk, that didn’t receive any shock compressed material serves as a reference without diamonds.

It is worth to mention that potentially, diamonds could be anywhere. As a pipe was pressed on the rim of the metal plate, the control samples are expected to have no or not many diamonds. If there are diamonds in this region, they likely stem from contamination during the transport of the metal discs.

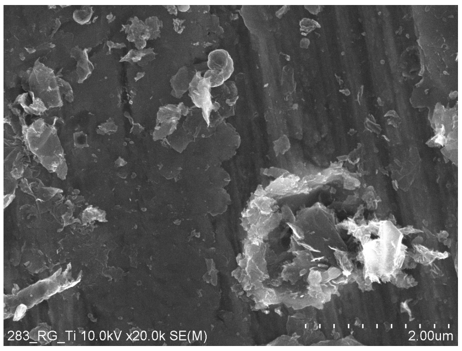

Figure 2: Example image from the diamond dataset

2.2.2 Pollen dataset

The pollen synthetic dataset [110] was created for testing machine learning models on segmentation or bounding box regression and classification tasks. The description of the dataset is taken from [110].

The pollen dataset counts 80.000 (resolution: 1280x1280 pixels RGB) images of "airborne pollen of different sizes within their natural range":

- 10.000 images *Chenopodium bonus-henricus* [29] making up the medium-sized pollen
- 10.000 images *Corylus colurna* [30] making up the medium-sized pollen
- 10.000 images *Urtica dioica* [114] making up the smaller pollen
- 10.000 images *Secale cereale* [102] making up the larger pollen
- 4x10.000 images each containing artefacts such as dust or burst pollen and a mix of pollen in the following ratios divided into 4 categories:
  - 'equal split' category contains: 25% *Chenopodium bonus-henricus*, 25% *Corylus colurna*, 25% *Urtica dioica*, 25% *Secale cereale*;
  - 'smaller pollen split' category contains: 70% Urtica dioica, 10% *Chenopodium bonus-henricus*, 10% *Corylus colurna*, 10% *Secale cereale*;
  - 'middle pollen split' category contains: 40% Corylus colurna, 10% Chenopodium bonus-henricus, 25% *Urtica dioica*, 25% *Secale cereale*;
  - 'bigger pollen split' category contains: 70% Secale cereale, 10% Urtica dioica, 10% *Chenopodium bonus-henricus*, 10% *Corylus colurna*

The data set is supported by an annotation set. It consists of:

- monochrome masks of each pollen slide (ignoring artefacts) for segmentation;
- x and y-coordinates of the bounding boxes containing all pixels of each of the pollen for regression;
- class names for each labelled pollen

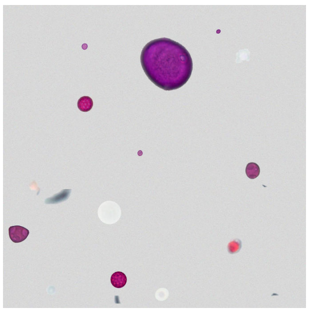

Figure 3: Example of image from the pollen dataset [110]

2.2.3 Geometric dataset

The Geometric dataset has been made especially for the purpose of the thesis. The Geometrical dataset was created for testing machine learning models on classification and localisation tasks.

This dataset was designed in a similar fashion as the real research dataset.

The Geometric dataset and the Diamond dataset were made on a similar concept of the spatial distribution of instances. Classes of datasets were made based on this distribution. At the same time ratio of instances in categories are similar to the pollen dataset.

It contains 90.000 (1280x1280 greyscale) images of geometric shapes (circles, triangles, rectangles) of different sizes:

- 15.000 images contain only Circles
- 15.000 images contain only Triangles
- 15.000 images contain only Rectangles
- 3x15.000 images contain a mix of geometric shapes in the following ratios divided into 3 categories:
  - 'Core' category has 15.000 images. It contains: Rectangles: 10%, Triangles: 10%, Circles: 80%. This is a simulation of the area with the highest density of objects of interest (circles).
  - 'Halo' category has 15.000 images. It contains: Rectangles: 45%, Triangles: 45%, Circles: 10%. This is a simulation of the area with the second-highest density of objects of interest (circles).
  - 'Control' category has 15.000 images. It contains: Rectangles: 50% Triangles: 50%. This is a simulation of the area with no objects of interest (circles).

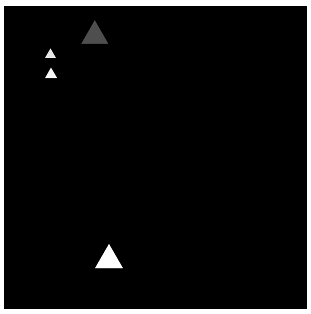
Figure 4: Example of image from the geometrical dataset []

3 Theory

In the following chapter, the basics that are necessary for the work are presented. A taxonomy of learning problems in machine learning is presented. The classification problem, Object recognition and localisation problem are presented in details. An overview of the methods of explainable AI and their classification is given. The details of backpropagation and perturbation based methods of explainable AI are given.

3.1 Introduction Deep Learning

The history of the *Deep learning* (DL) starts in 1967 [123]. The first learning algorithm was published by Alexei Grigorievich Ivakhnenko and Valentin Grigorevich Lapa in the paper Cybernetics and Forecasting Techniques [43]. The article was about controlled feed-forward deep multilayer perceptrons. After that, in 1971 A. G. Ivakhnenko described a network trained by batch processing [49]. Other working architectures for machine learning, architectures built for computer vision, began their history with the ”Neocognitron” presented in 1980 by Kunihiko Fukushima. [36]. The term “deep learning”, in turn, was presented to the world by Rina Dechter in 1986 [24] [25] and artificial neural networks were introduced by Igor Aisenberg and colleagues in 2000 [11]. 2012 was marked by the win of the ImageNet competition by the system based on the Convolutional Neural Network (CNN) [61], [20]. In the same year, a multitasking deep neural network won the ”Merck Molecular Activity Challenge” [21]. This year was named later as the beginning of the “deep learning revolution” [105].

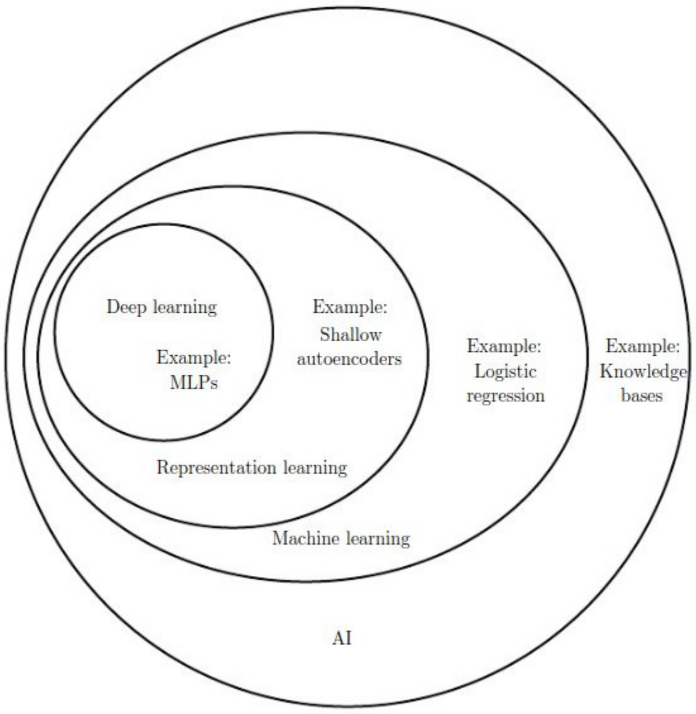

Figure 5: Connection of terms AI, ML, DL [40]

Briefly, the term Machine learning (ML) can be described as "algorithms that improve automatically through experience" [81]. Deep learning is a group of tasks that is a part of machine learning. The visualization of the classification can be found in Figure 5. This work focuses on deep learning topics.

Deep learning extracts high-level features from raw input using a variety of sequential nonlinear transformations organized in multiple layers; this structure is usually represented as artificial neural networks [67]. In this approach, successive transformations of the input data are produced until a final transformation predicts the output.

Most modern approaches to deep learning are based on artificial neural networks (NN). Neural networks form the basic building blocks - artificial neurons named Perceptrons (inspired by biological neurons). The structure of which can be represented as:

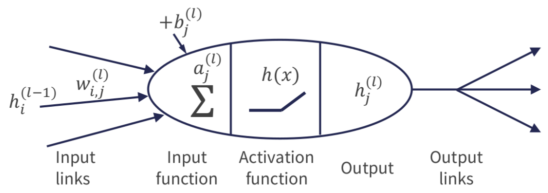

Figure 6: One Perceptron [44]

In Figure 6: h_i^{l-1} is the input to the unit in layer l; $w_{i,j}^l$ is the weight on link from unit i in layer $l-1$ to this unit; a_j^l is a linear combination of the input; b_j^l is a bias value; $h(x)$ is the output (input to next unit), e.g. logistic, ReLU [124], [44].

Neurons are organized in layers. The word "deep" in the "deep learning" term is connected with the number of layers in the model through which the data is converted. Figure 7 presents an example of a neural network with one hidden layer (not directly observable from the model inputs and outputs). The number of layers can be several hundred [103]. Having more than one hidden layer allow neural networks to learn data representations consist of multiple levels of abstraction.

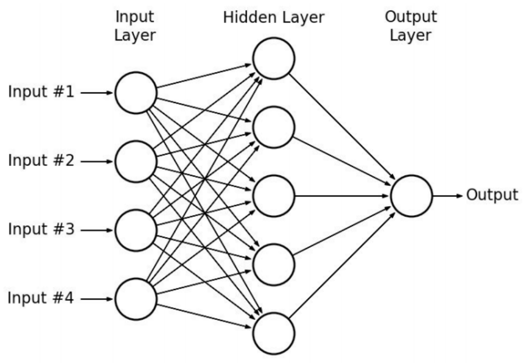

Figure 7: The Neural network with hidden layer [75]

Based on the type of data available and the studying research question, the scientist will select a learning algorithm using a specific learning model.

There are three types of machine learning:

1. Supervised Learning (see 3.1.1)
2. Unsupervised Learning (often referred as part of Semi-supervised and unsupervised problem learning)
3. Reinforcement Learning [56]

Supervised learning

1. **Regression** task - forecast based on a sample of objects with different characteristics. Regression is the problem connected with approximating a function of transformation input variables to a continuous output variable. Continuous output variables are a real-values [17].
2. The task of **classification** is to predict outputs (belonging to one or another category) based on a set of features. There are a finite number of outputs. The task of approximating a function of mapping from input variables to discrete output variables is a classification. The output variables are labels (categories). The required function probabilistically predicts the class or label for observation (the instance from the dataset).

Unsupervised Learning

1. **Clustering**. Algorithms that learn hidden patterns from unlabelled data named Unsupervised learning methods [118]. Clustering, which is part of unsupervised learning, groups instances based on their similarity without using

class labels, that is, the task of clustering is to distribute data across a number of groups.

2. The purpose of **anomaly detection** is to separate anomalies from standard cases. At first glance, this coincides with the task of classification, but there is one significant difference: anomalies are rare, and the training examples on which you can train a machine learning model to identify such objects are small. In practice, such a task is, for example, the detection of fraudulent activities with credit or debit cards.

3. **Dimensionality reduction** is a method for reducing the number of input functions (variables) in a dataset. A large number of input functions makes the prediction task more difficult for the model. The reduction of dimensions can be used to visualize data [90].

4. Approaches for learning **latent variable models**. Observed data of these models were generated by unknown latent variables. Models define the opportunity to clarify prior knowledge and structural relationships in complex datasets. A common case is when the latent variables predict the mean of observations. It can be used in Natural Language processing. [58], [12].

It is important to mention that modern research in Deep Learning is not limited to the problems listed above. There are Hybrid problems, for example:

1. Self-Supervised Learning (see 3.1.2)

2. Semi-Supervised and Unsupervised Learning

3. Multi-Instance Learning [18]

The majority of machine learning tasks fall into one of the following categories and frequently has a link to more general tasks mentioned above.

Due to the context and task of the thesis only supervised learning will be considered. Among hybrid learning problems, Self-Supervised Learning receives closer scrutiny as a main approach of the thesis. The mathematical foundation of this chapter is based on [13].

3.1.1 Supervised learning

Supervised learning can be formulated informally as the problem of finding in family $g : \Theta \to Y^X$ of functions, one $g : X \to Y$ that minimizes a weighted sum of two objectives:

1. g deviates little from a finite set $\{(x_s, y_s)\}_{s \in S}$ of input-output-pairs

2. g has low complexity, as quantified by a function $R : \Theta \to \mathbb{R}_0^+$

Where $S \neq \emptyset$ finite. S has a name set of samples.

It is known that the family g can have meaning beyond a simple parameterization of functions from X to Y. For example, Θ can be a set of forms, g are functions defined by these forms, and R the length of forms. Then, supervised learning is actually a problem of optimizing over forms of functions, and R penalizes the complexity of these forms. Furthermore, g can be chosen so as to primarily restrict the set of functions from X to Y.

Focus of the attention will be concentrate exclusively on the special case when Y is finite. For the simplicity assume Y $= \{0,1\}$.

Optimize over a family f : $R : \Theta \to \mathbb{R}^X$ and defining g w.r.t. f via a function $L : \mathbb{R} \times \{0,1\} \to \mathbb{R}_0^+$ called a loss function, such that

$$\forall \theta \ \in \Theta \ \forall x \in X : \ g_\theta(x) = \operatorname*{argmin}_{\hat{y} \in \{0,1\}} L(f_\theta(x), \hat{y}) \tag{1}$$

The equation 1 and its derivation was formulated in this way in [13], reproduced here.

This approach performs sequential transformations of the input data until the final transform predicts the output. These transforms are derived from predefined input-output pairs so that the network knows from examples of how the transforms should be performed.

For the supervised learning classification task, it is necessary to identify which of a set of labels a new observation belongs to. It can be done based on a training set of data containing instances where the labelling was made.

3.1.2 Self-Supervised learning

Self-supervised learning is a way that virtually unlimited labels can be generated from existing images and to use these labels to learn the representations [59] [53].

Thus, instead of human annotations, creative exploit of some property of data is used (to set up a pseudo-supervised task).

As far as the representation is learned, it is possible to use transfer learning to tweak it for some supervised tasks such as classification or localisation (*downstream task*).The downstream task is used to evaluate the quality of features learned in the self-supervised learning [53].

In comparison with supervised learning methods that require a data pair $X_s = f_\theta(x_s)$ and y_s while y_s is annotated by humans, self-supervised learning trained with data X_s as well at the same time with its *pseudo* label P_s. P_s is automatically generated for a pre-defined pretext task involving zero human annotation. Attributes of images or videos can be used for a generation the pseudo label P_s (for example the context of images), or by traditional hand-designed methods.

Many self-directed learning techniques have been developed for learning visual features with not using human-annotated labels.

$D = \{P_s\}_{s \in S}$ is a set of N training data. The training loss function is defined as:

$$\min_{y \in Y} \inf_{\theta \in \Theta} \lambda R(\theta) + \frac{1}{|S|} \sum_{s \in S} L(f_\theta(x_s), p_s) \tag{2}$$

The methods can be marked as self-supervised learning so long as pseudo labels P are generated automatically without involving human annotations. The main focus of this thesis will be on the self-supervised learning method used developments other methods designed for visual feature learning. Of particular interest is the fact that the features of the self-supervised learning methods can be carried over to multiple visual tasks and perform new tasks by exploring limited labelled data [53].

The classification about the self-supervised pretext tasks was presented in [53]. It can be presented as:

1. Generation-based Methods: Techniques of this category study visual features by solving pretext tasks that involve the creation of images or videos.

   Visual features are learned during the imaging process. The group of methods: image colorization [129], image super resolution [64], image inpainting [89], image generation with Generative Adversarial Networks (GANs) [41], [131].

2. Context-based pretext tasks: When developing pretext contextual tasks, the contextual features of images or videos are mainly used, such as contextual similarity, spatial structure, temporal structure.

   (a) Context Similarity: Pretext tasks are mainly designed on the similarity of the context between fragments of images. The method include: image clustering based methods [85], graph constraint-based methods [66].

   (b) Spatial Context Structure: Convolutional neural networks based on spatial relationships between image fragments can be trained by Pretext tasks. This type of methods includes: image jigsaw puzzle [84] [57], context prediction [27], and geometric transformation recognition [54], [37].

   (c) Temporal Context Structure: ”The temporal order from videos is used as supervision signal” [80], [65].

3. Free Semantic Label-based Methods: Generated automatically semantic labels can be used for training neural networks by these tasks. The labels can be generated by traditional algorithms [32], [117] or by game engines. The part of the group are: moving object segmentation [88], [62], contour detection [97], [47], relative depth prediction [52], and etc.

4. Cross Modal-based Methods: This type of pretext tasks trains NN to check the match of two different channels of input data. The methods include: Visual-Audio Correspondence Verification [14], RGB-Flow Correspondence Verification [101] and egomotion [51].

3.2 Classification

Labelled data is required for classification. There is a training sample in which objects are presented in the form of their feature description (feature vector) and class label. It is necessary to find an algorithm for each new object (its attribute description) that will define the class label of this object. This is equivalent to building a dividing surface in a multidimensional feature space.

There are different methods for the classification task:

- Logistic Regression;
- Decision Tree Algorithm;
- k-Nearest Neighbor (KNN);
- Artificial Neural Networks and Deep Neural Networks;
- and many other methods.

According to the task of the thesis, classification using deep neural networks is of interest. The learning problem can be written as follows. The formulation of the learning problem in this way was presented in [13], reproduced here.

For any finite set $A \neq \emptyset$ the elements of which should be classified and any finite set $B \neq \emptyset$ of class labels, there is a special interest in *maps* $\varphi : A \to B$ that assign to every element $a \in A$ exactly label of one class $\varphi(a) \in B$. Maps are exactly those subsets of $\varphi \subseteq A \times B$ that satisfy

$$\forall a \in A \; \exists b \in B : \quad (a, b) \in \varphi \tag{3}$$

$$\forall a \in A \; \forall b, b' \in B : \quad (a, b) \in \varphi \wedge (a, b') \in \varphi \Rightarrow b = b' \tag{4}$$

They are characterized by functions $y : A \times B \to \{0, 1\}$ that satisfy

$$\forall a \in A : \sum_{b \in B} y_{ab} = 1 \tag{5}$$

The problem of learning and inference maps have to be reduced to a problem of learning and inference of solutions by choosing limited data with

$$S = A \times B \tag{6}$$

$$Y = \Big\{ y : A \times B \to \{0, 1\} \mid \forall a \in A : \textstyle\sum_{b \in B} y_{ab} = 1 \Big\} \tag{7}$$

Let's consider some finite set $V \neq \emptyset$ and constrained data (S, X, x, Y) with $S = A \times B$ as in 6, $X = B \times \mathbb{R}^V$ and Y as in 7. More specific, let's assume that, for any $(a, b) \in A \times B$, the class label b is the first attribute of (a, b), i.e.,

$$\forall a \in A \; \forall b \in B : \exists \hat{x} \in \mathbb{R}^V : \quad x_{ab} = (b, \hat{x}) \tag{8}$$

In a special case, this is labeled data, in which only one $Y = \{y\}$ is given with y satisfying the constraints 6 [13].

Regarding linear functions, more specifically: $\Theta = \mathbb{R}^{B \times V}$ and $f : \Theta \to \mathbb{R}^X$ such that

$$\forall \theta \in \Theta \; \forall b \in B \; \forall \hat{x} \in \mathbb{R}^V : \quad f_\theta((b, \hat{x})) = \sum_{v \in V} \theta_{bv} \hat{x}_v = \langle \theta_b, \hat{x} \rangle \tag{9}$$

For the formulation of the learning problem the Random variables (formulated in [13] and reproduced here) should be denoted.

Random variables

- For any $(a, b) \in A \times B$, let X_{ab} be a random variable whose realization is a vector $x_{ab} \in B \times \mathbb{R}^V$, called the *attribute vector* of (a, b)
- For any $(a, b) \in A \times B$, let Y_{ab} be a random variable whose realization is a binary number $y_{ab} \in \{0, 1\}$, called the *decision* of classifying a as b
- For any $b \in B$, and $v \in V$ let Θ_{bv} be a random variable whose realization is a real number $\theta_{bv} \in \mathbb{R}$, called a *parameter*
- Let Z be a random variable whose realization is a subset $z \subseteq \{0, 1\}^{A \times B}$.

Learning problem

The lemma formulated in [13] are reproduced here, recalling here only what is necessary, without the proof.

Lemma 1 *Estimating maximally probable parameters θ, given attributes x and decisions y [13], i.e.,*

$$\underset{\theta \in \mathbb{R}^{B \times V}}{\operatorname{argmax}} \; p_{\Theta|X,Y}(\theta, x, y) \tag{10}$$

is identical to the supervised learning problem w.r.t. L, R and λ such that

$$\forall r \in \mathbb{R} \; \forall \hat{y} \in \{0, 1\} : \quad L(r, \hat{y}) = -\hat{y} r + \log(1 + 2^r) \tag{11}$$

$$\forall \theta \in \Theta : \quad R(\theta) = ||\theta||_2^2 \tag{12}$$

$$\lambda = \frac{\log e}{2\sigma^2} \tag{13}$$

Moreover, this problem separates into $|B|$ independent supervised learning problems, each w.r.t. parameters in $\mathbb{R}^V$, with L and λ as above, and with

$$\forall \theta' \in \mathbb{R}^V : \quad R'(\theta') = ||\theta'||_2^2 \tag{14}$$

Analyzing the above formulation of the classification problem as an instance of a supervised learning problem and given training data consisting of pairs of input-output values the task of the classification can be written as follows. The task is to "analyze" the training data and create an estimated function that can be used to classify new instances to which the network did not have access by assigning the most likely class label to each one. Thus, the outputs of the model are probability vectors.

3.3 Explainable AI with extremal perturbation

Deep learning (DL) algorithms can collect and process large amounts of data. It showed success in competitions such as Imagenet [61]. DL models can learn complex patterns that allow them to make predictions about data not included in the training. At the same time, DL algorithms are complex and difficult to understand. Many neural networks are not designed to be interpretable. Lack of transparency and accountability of models used in vast numbers of areas can have serious consequences due to incorrect usage and inability to justify the models and results. This section in many ways based on [38] and [33].

There are a need and interest among the research community for clarifying the basics of deep learning predictions and a more intuitive understanding of results from deep learning networks.

Explanations of work and focus of deep networks can be divided into two groups: explaining the *processing* of data by the network and explaining the *representation* of data within the network [38].

Explanations of Deep Network Processing. It also referred to as the Attribution approach. This is the main approach for explainable AI. It includes methods which are aiming to reduce the complexity of the neural network or add some minor changes into the network to see the results of the experiment and based on this make conclusions about the effects of the network. In other words which parts of the input of the network are the most responsible for the output. Examples are:

- Approximation-based methods. The approach is illustrated by LIME method [98] and [112]. In these models, the black box system is explained by analyzing the disturbance behaviour of the input data, and then the data is used to build a local more simple linear model that using as a downgraded representation for the full model.
- Backpropagation methods. These are attribution techniques that use back-propagation to track information from the output of the network back to the input, or a middle layer.
- Perturbation methods. For these methods inputs of the model is being perturbated and to observe changes in the output. It can be implemented by occlusion

patterns, optimization of spatial perturbation mask, or by perturbations of the input.

- Visualizations of intermediate activations. To characterize the behaviour of the filter the algorithm learns dataset examples from the training set that maximally activate the filter. In addition, there is an approach that studies the image, which reconstructs the intermediate network activations using the natural image to visual clarity.
- Automatic-Rule Extraction. An automatic rule extraction is an approach for summarizing decisions. The approach supposes the extraction of rules. Examples of this approach are the FERNN [109], KT method [87] etc.
- Decision trees [122]. It is a simple algorithm based on a sequence of decisions. The decisions made according to Information Gain [122].

Explanations of Deep Network Representations. The explanation of deep network representations has a goal to understand the structure and role of the data flowing through "bottlenecks". It can be done on different stages, where the network learns:

- Layers
- Individual Units
- Representation Vectors

Explanation-Producing Systems. The aim of the method is to create a network that is designed to be easier to explain. There are several different approaches:

- Attention Networks
- Disentangled Representations
- Generated Explanations

The most interest in the context of the thesis has two methods from "Explanations of Deep Network Processing" group: Backpropagation-based methods and Perturbation based methods.

3.3.1 Backpropagation-based methods

This kind of approach often has an aim to obtain the map showing which parts of the input data actually have an influence on the network output (salience map). Some authors group these approaches in the "salience mapping" category. In some methods, a salience map can be created by directly computing the input gradient [112]. It uses unmodified backpropagation and visualizes the derivative of the network's output. Another approach is (e.g., Guided Backprop [115], and SmoothGrad

[113]) reduce the noise in the gradient signal by fine-tuning the backpropagation rules of certain layers.

Because derivatives can miss important features of the information that pass through a network, some of the approaches have been designed to propagate quantities other than gradients. Such methods include CAM [130], GradCAM [107]. These methods use combining gradients, weights of the network and activations at certain layers.

Grad-CAM The Grad-CAM method is based on the CAM [130] method. CNN architecture is modified using the CAM approach. Fully-connected layers of the CNN architecture was replaced by convolutional layers and a global average pooling layers [68]. Thereby it achieves class-specific feature maps. The Grad-CAM method combines feature maps using the gradient signal which makes not necessary the neural network architecture modifications. This makes it possible to apply the approach to ready-made architectures based on CNN [107].

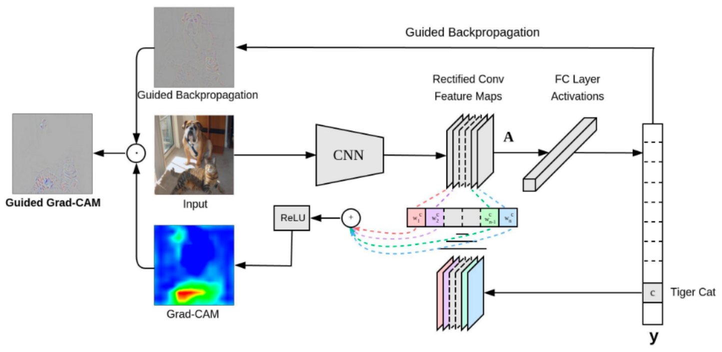

Figure 8: Grad-CAM architecture [10]

Grad-CAM uses gradient information supplied to the final CNN convolutional layer for importance assignment values for every neuron for a specific solution of interest.

To obtain a localization map with class discrimination $L^y_{Grad-CAM} \in R^{u \times v}$ of width u and height v, Grad-CAM computes the gradient g_y (which is a score for class Y) relative to the A^k. They are feature map activations of the convolutional layer. To obtain importance weights α^y_k gradients are combined into a global average-pooled for width and height. The index i is for the width dimension and j is for the width dimension [107]:

$$\alpha^y_k = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \qquad \underbrace{\frac{\partial g^y}{\partial A^k_{ij}}}_{\text{gradients via backprop}} \tag{15}$$

According to the Grad-CAM authors [107]: "Weight α^y_k shows a partial linearization of the deep network downstream from A, and captures the 'importance' of feature map k for a target class y".

Heatmap of Grad-CAM is a weighted combination of activation maps followed by a RelU function:

$$L^y_{Grad-CAM} \in R^{u \times v} = ReLU \underbrace{\left(\sum_k \alpha^y_k A^k \right)}_{linear\ combination} \tag{16}$$

These methods are based on trivial modifications of the backpropagation algorithm. Thereby, these methods are efficient. However, except for the explicit advantages of these methods, there are some disadvantages. Analysis of publications [74] [8] [33] have shown that methods can "see" average network features but in some cases not able to characterise intermediate activations or individual outputs. The authors of the following papers come to similar conclusions. More over that modern researches show that commonly used saliency map approaches less trustworthy than previously thought [15] [26]. Saliency map methods fail at least once in conducted experiments[15]. The authors recommend use detection or segmentation models for the localisation task instead of saliency maps in the high-risk domain of medical imaging.

3.3.2 Perturbation-based methods

These techniques amount to selectively deleting (or preserving) portions of the input data and noticing the effect of this makes on the model output. The advantage of the approach is that the value of analysis is clear from the beginning.

Perturbation method can be formulated as it was done in [33]: Let $x : \Omega \to \mathbb{R}^3$ be a colorful image where $\Omega = \{0, ..., H-1\} \times \{0, ..., W-1\}$ is a discrete lattice. Φ is a model, e.g. CNN. It maps the image to a scalar output value $\Phi(x) \in \mathbb{R}$ [33].

These methods study which part of x excite the model causing the response $\Phi(x)$ to be large. Par excellence, it is necessary to find a mask m assigning to every pixel $u \in \Omega$ a value $m(u) \in \{0, 1\}$, where $m(u) = 1$ means that the pixel maximally contributes to the output. A value $m(u) = 0$ relates to no contribution at all.

The importance of a pixel can be evaluated as follows: the mask should be used to induce a local perturbation of the image: $\hat{x} = m \otimes x$. All pixels with $m(u) = 1$ are saved, while other pixels are blurry. The aim here is to find a small subset of pixels that are (when stored) enough to store a large output value $\Phi(m \otimes x)$. In the frame of the article [34] authors use the following approach to identify salient pixels by solving an optimization problem:

$$m_{\lambda,\beta} = \operatorname*{argmax}_m \Phi(m \otimes x) - \lambda ||m||_1 - \beta S(m). \tag{17}$$

The first part of the equation 17 promotes to increase in the response of the network. The second part forces the mask to highlight a small part of the input image, blurring other pixels. The third part is responsible for the smoothness of the mask [33].

How correctly noted the authors of the articles [99], [33]: there is a problem with the formulation presented in equation 17. Terms presented in the equation are not rateable. The choice of different λ and β values in equation 17 will result in different masks. It is not possible to compare them properly.

Extremal perturbation The solution of the issue mentioned in equation 17 was found by authors of [33]. They have presented the constraint for the area of the mask to make it a fixed value (as a fraction $a|\Omega|$ of the input image area). Also, they propose to control the mask smoothness by choosing it in a fixed set of smooth functions M.

The mask that maximizes the model's output was presented in [33]:

$$m_a = \underset{m:\, ||m||_1 = a|\Omega|,\, m \in M}{\operatorname{argmax}} \Phi(m \otimes x). \tag{18}$$

It is important to say that for the chosen area a the resulting mask is a function of a only. The concept of extremal perturbation (EP) was defined as follows. Φ_0 is a a lower bound on the model's output. The next step is to find the "smallest mask that achieves at least this output level"[33] (same as changing the parameter a in equation 18):

$$a^* = \min \left\{ a : \ \Phi(m_a \otimes x) \geq \Phi_0 \right\}. \tag{19}$$

The mask m_a^* is extremal because saving a portion smaller then this from the input image is not enough to trigger a network response above Φ_0.

A single extremal mask is informative since it characterizes a family of input perturbations. [33]. This makes extremal perturbations similar by a concept to methods like [98], [34], which analyzing input-output mapping, for example, the gradient [112] and LIME [98].

To define area constraint it is necessary to optimize equation 18 by a gradient-based method, authors relax the mask to span the full range [0, 1]. One of the potential approaches, in this case, is to count a number of $m(u)$ values which are close to the value 1 and penalize masks in case the count is different from the target value $a|\Omega|$. To do this authors propose a *vecsort* vector which is contains vectorised and sorted values in non-decreasing order. $vecsort(m) \in [0,1]^{|\Omega|}$. The output of $vecsort(m)$ is a vector $r_a \in [0,1]^{|\Omega|}$ if the mask m satisfies the area constraint exactly. The vector is consisting of $(1-a)|\Omega|$ zeros followed by $a|\Omega|$ ones. The regularization term: $R_a(m) = ||vecsort(m) - r_a||^2$. The equation 18 takes the form:

$$m_a = \underset{m \in M}{\operatorname{argmax}} \Phi(m \otimes x) - \lambda R_a(m). \tag{20}$$

In addition, to solve optimization issues the authors of [33] developed the new max-convolution operator and smooth max operator for the smoothing of the extremal perturbation mask.

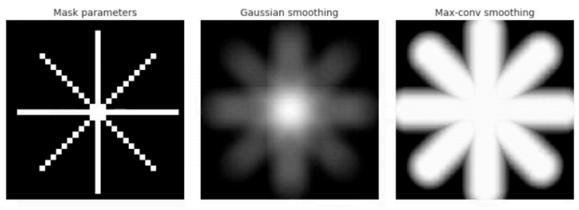

Figure 9: Convolution operators for smooth masks [33]

The comparison of the extremal perturbation method with the attribution methods shown in Figure 10.

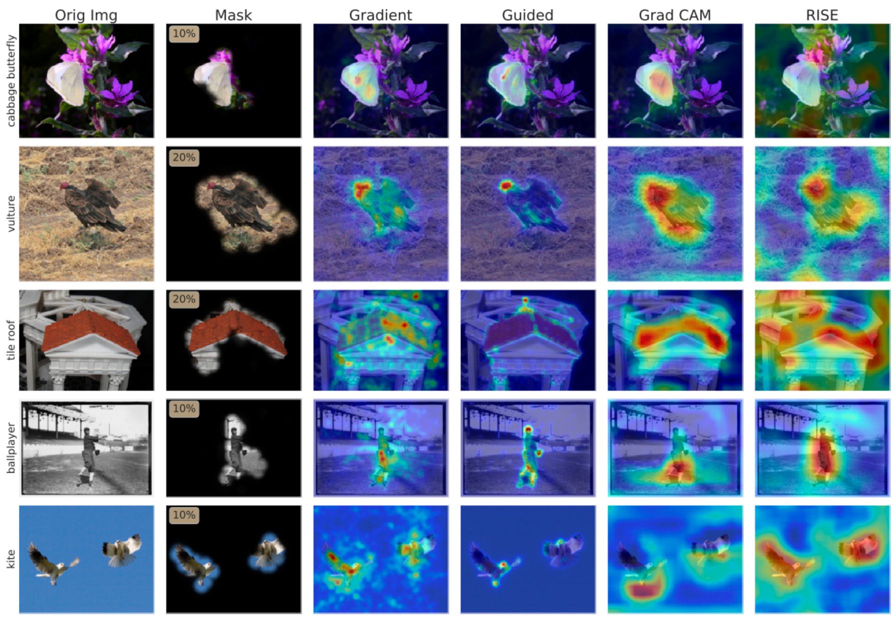

Figure 10: Comparison of attribution methods [33]. Comparison of the extremal perturbations (optimal area a in box) to several popular attribution methods: gradient [112], guided backpropagation [115], Grad-CAM [107], and RISE [91]

The standard approach to evaluate attribution methods is the Pointing game. The method assumes to correlate semantic annotations in images with the output of methods. The saliency map for each of the object classes presented in the image has to be computed by the attribution method. "One gets hit if the maximum point on the saliency map is contained within the object" [126]. The overall accuracy contains the number of hits versus the number of hits and misses.

The results of the evaluations of the extremal perturbation (EP) method on the PASCAL and COCO datasets is competitive with other methods. See the Table 1

Table 1: Pointing game. Mean accuracy on the pointing game over the full data splits (All) and a subset of difficult images (Diff). Results from PyTorch re-implementation using TorchRay package [33]

	VOC07 Test (All/Diff)	VOC07 Test (All/Diff)	COCO14 Val (All/Diff)	COCO14 Val (All/Diff)
Method	VGG16	ResNet50	VGG16	ResNet50
Cntr	69.6/42.4	69.6/42.4	27.8/19.5	27.8/19.5
Grad	76.3/56.9	72.3/56.8	37.7/31.4	35.0/29.4
DConv	67.5/44.2	68.6/44.7	30.7/23.0	30.0/21.9
Giud	75.9/53.0	77.2/59.4	39.1/31.4	42.1/35.3
MWP	77.1/56.6	84.4/70.8	39.8/32.8	49.6/43.9
cMWP	79.9/66.5	**90.7**/82.1	49.7/44.3	**58.5/53.6**
RISE	86.9/75.1	86.4/78.8	50.8/45.3	54.7/50.0
GCAM	86.6/74.0	90.4/ **82.3**	**54.2/49.0**	57.3/52.3
Exremal perturbation	**88.0/76.1**	88.9/78.7	51.5/45.9	56.5/51.5

The approach of extremal perturbation analysis avoids some of the issues of prior work in this area and can be useful for the explanation of the basics of the decision of Deep neural networks.

3.4 Object Recognition and Localisation

Recognition and localization of visual images are some of the most important components for modern information systems, automatic systems and systems for making decisions. Issues bounded with the identification of objects and signals, characterized by a set of certain properties and characteristics, originate in such industries as robotics, autonomous vehicles driving, search for information, analysis and monitoring of visual data, and artificial intelligence research.

Object recognition is a common term to explain a cluster of computer vision tasks that connected with the identification of objects in images. Object localization and object detection are popular tasks of computer vision [72].

Localisation : Identifying the position of the object by educting the object by making a frame. The frame has the name "bounding box".

Input: The input contains several objects located on one image (for example a photograph).

Output: Bounding boxes, one or several of them (e.g. defined by coordinates).

Object detection: Classification and detection of all objects in the image. It also means the assignment of a class label to every object and the creation of a bounding box for every object. Thus, object detection combines classification and localisation tasks.

Input: The input contains several objects located on one image (for example a photograph).

Output: Bounding boxes, one or several of them (e.g. defined by coordinates), plus a class label for every detected object (bounding box).

The localisation task is a version of the object recognition problem. The formulation of the task is limiting the tasks to objects with the same type within an image. These two tasks have a close connection and the same deep learning models are using.

In the next sub-section, Convolutional Neural Networks (CNNs) will be considered, which are used for the object localisation and object detection tasks.

Convolutional Neural Networks Neural networks based on multilayer perceptrons were the standard approach for imaging before CNN. CNN solves this problem by taking 2D image topology into account and using a new type of architecture. CNN is made up of different types of layers: connected layers, pooling layers, convolutional layers. Convolutional neural networks provide partial resilience to changes in scale, displacement, rotation, change of perspective, and other distortions. The example of CNN work was presented in Figure 11. ReLU function was used as the activation function for this example.

$$ReLU(x) = \max(0, x). \tag{21}$$

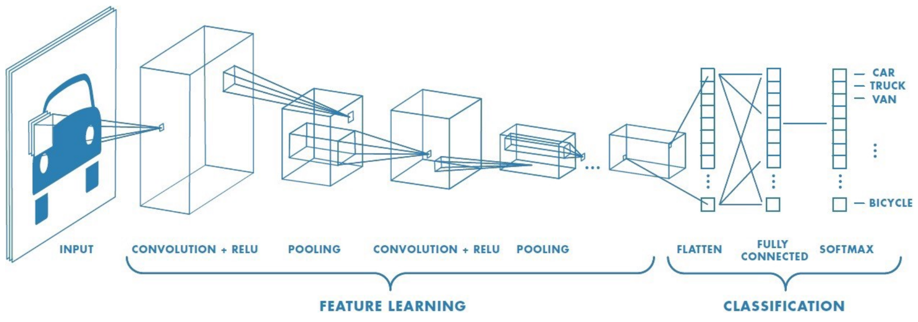

Figure 11: Sketch of a CNN classify images [100]

The **Convolutional Layer** has a set of trainable filter masks. The size of these filters is usually small. The filter size is usually taken in the range from 3x3x3 to 7x7x3 (the last number is equal to the depth of the input). If the size then it will not be able to distinguish any signs, if it is too large, then the number of connections between neurons increases.

The filter mask slides over the entire area and is applied to each pixel of the input image and finds certain features of the objects. These filters can be thought of as detectors of certain visual characteristics.

To reduce the number of parameters and therefore the computation the **Pooling layer** performs a reduction of the spatial size. Usually, the maximum of a 2x2 area is taken to reduce the size which means that 75% of the activations are discarded. Figure 12 is illustrating this process.

The next step after the convolution is the **Pooling layer**. It takes as input small, separate pieces of the image (usually 2x2) and combines each piece into a single value. Several aggregation methods are possible, most often a maximum of four pixels is selected. The layer can reduce the number of parameters, which will lead to fewer calculations. This can be thought of as downsampling.

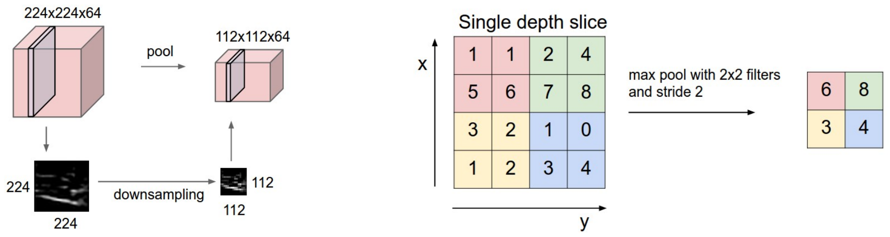

Figure 12: Pooling layer operation [55]

After a series of convolutional and pooling layers, the next step is **fully-connected layers** with a softmax layer as the final layer. It takes each pixel as an independent value. The softmax function turns a vector of real numbers into a vector of probabilities. The output values are the probabilities of the image falling into a particular class.

Architectures for localisation and object detection Modern object detectors are based on a proposal-based two-stage mechanism. The R-CNN framework family [39] is one of the most popular architectures today. The family includes the R-CNN, Fast R-CNN, and Faster-RCNN. In the first step, a set of potential object locations is created, and in the second step, every candidate location is classified as one of the foreground classes or as background classes by a CNN 3.4. Thanks to a number of improvements [96] [70], this two-stage structure consistently provides the best accuracy in the complex COCO test [69]. This paragraph in many ways based on [71].

There are several types of architectures used in object detectors:

- Classic Object Detectors. The paradigm in which the classifier is using a dense image grid has the name a sliding window paradigm.

  One of the first papers regarding the sliding-window approach was the work of LeCun et al. [63]. Through a sequence of advances such as HOG [22] and ”integral channel features” [28] it was the leading detection method in computer vision, with the grow of deep learning [60], however two-stage detectors very soon started to be a main object detection method.

- Two-stage Detectors. This is the main method in the area of modern object detection. ”In the first step, a set of candidates is created that must contain all objects while filtering out most of the negative locations, and in the second step, the candidates are classified” [71]. R-CNN [39] made an upgrade the second-stage classifier to a convolutional network. R-CNN had improvements such as using "learned object proposals" [92] [96], increasing speed of work [45], [127]. One of the most significant improvements under the R-CNN network was the integration of Region Proposal Networks (RPN) with the second-stage classifier into a single convolution network creating the Faster RCNN.

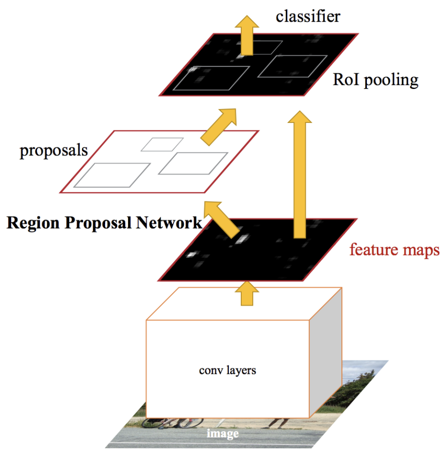

Figure 13: Summary of the Faster R-CNN Model Architecture. Taken from: Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks [96].

- One-stage Detectors. One-stage detectors are used to regularly and tightly sample the locations of objects. One of the first deep networks object detectors with one stage was OverFeat [108]. SSD [73] [35] and YOLO [94] [95] are examples of one-stage detectors. Both of them are focused on speed. YOLO and SSD have an accuracy of around 10-40%. The accuracy of these methods is relative to state-of-the-art two-stage methods [71]. Two-stage detectors can be faster by reducing the resolution of the input image and the number of the proposals, but one-stage methods are inferior in accuracy even with a bigger compute supply [48]. Despite this, one of the best results among one-stage detectors shows Retinanet architecture [71].

RetinaNet is a unified single network consisting of a backbone network and two sub-networks designed for specific tasks. The backbone computes a convolutional feature map for the input instance (image). The backbone is an autonomous convolutional network.

The first subnet makes the convolutional classification of objects at the output of the backbone. The second subnet is responsible for the convolutional bounding box regression. The two mentioned subnets have a special design created for the specific task of one-stage dense detection. It is presented in Figure 14.

Focal Loss (FL) has been presented also in [71]. If during training there is an extreme imbalance between the foreground and background classes the focal loss can be used. It is an important part of RetinaNet architecture.

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t) \tag{22}$$

where $p_t \in [0, 1]$ for the class label $y = 1$ it is a model's estimated probability.

$$p_t = \begin{cases} p & if\ y = 1 \\ 1 - p & otherwise \end{cases} \tag{23}$$

Thus the Focal Loss adds a factor $(1 - p_t)^\gamma$ to the standard cross-entropy criterion. The Cross Entropy (CE) in turn can be formulated as "the average number of bits needed to encode data coming from a source with distribution p when we use model q" [83]. Setting $\gamma > 0$ decreases the loss for correctly classified examples ($pt > 0.5$), putting more focus on difficult, misclassified examples. When $\gamma = 0$, authors make a conclusion that Focal Loss is equivalent to Cross Entropy. $\gamma = 2$ works in practice and the RetinaNet is comparatively sensible to $\gamma \in [0.5, 5]$[71].

The focal loss adjusts the class imbalance. This factor is important for single-stage detectors. It gives an opportunity to effectively training on different kind of examples. The examples can be used without sampling as well as without negatives suppressing losses and computed gradients.

RetinaNet contains the following parts:

Feature Pyramid Network Backbone. Authors adopt the Feature Pyramid Network (FPN)as the backbone network for RetinaNet. Originally it was presented in [70]. "FPN augments a standard CNN with a top-down pathway and side connections to the network" [70]. Thus authors build FPN on top of the ResNet architecture.

Anchors: Authors use translation-invariant anchor boxes corresponding with what was used in the Region Proposal Networks (RPN) [96] variant in [70].

At the same time, every anchor was appointed one-hot vector of classification targets of length K. K is the number of object classes and the 4-vector of the regression target box.

Classification Subnet: It predicts the probability of the potential event in which an object being present at every position for every and object classes (K), anchors (A) . This subnet represented by a minor Fully convolutional network attached to every FPN layer.

Box Regression Subnet: In addition to the object classification subnet, authors attach minor FCN to every level of the pyramid. "The creation of Box Regression Subnet aims to regress the offset from every anchor box to a nearest "ground-truth object"" (if it exists)[71].

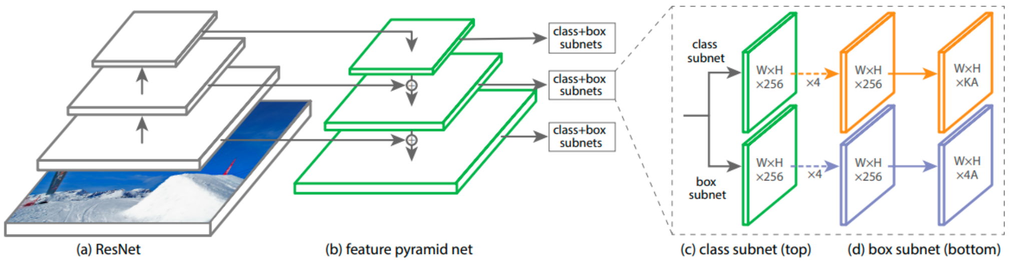

Figure 14: The one-stage RetinaNet network architecture [71]

As was written before, RetinaNet uses a Feature Pyramid Network (FPN) [70] backbone on top of ResNet architecture [46]. It was presented in Figure 14. According to figure: (a) This is ResNet architecture. FPN stage (b) has the aim of generating a rich multiscale pyramid of convolutional functions. This is a backbone. Classification Subnet (c) (for classifying anchor boxes) and Box Regression Subnet (d) (for regressing from anchor boxes to "ground-truth object" boxes) have been attached to this backbone.

According to the authors, the network was designed is deliberately simple to focus on a new focal loss function that closing the "distance" in accuracy between the presented single-stage detector and modern two-stage detectors such as the Faster R-CNN with FPN, while operating at higher speeds [71].

Results of the comparison ReinaNet architecture on the challenging COCO dataset in comparison with both one-stage and two-stage models are presented in [71] as well. The COCO dataset has its own detection evaluation metrics [69].

The main terms of the COCO-metrics presented as follows. IoU is defined as the area of intersection of the predicted bounding box and ground truth box divided by the area of the union of them. The main metrics are Average Precision (AP) and Average Recall (AR), where the precision can be formulated as a positive predictive value and recall as a true positive rate. AR is the recall averaged over all IoU. AP is the area under the precision-recall curve (it was also averaged across all classes of the dataset).

Compared to other one-stage methods, the RetinaNet architecture (with ResNet-101) achieves in the Average Precision metric: 39.1 vs. 33.2 of the DSSD [35] which has obtained the closest result. Compared to modern two-stage methods, RetinaNet (with ResNeXt-101) achieves in the Average Precision metric 40.8 in comparison with the best performing Faster R-CNN model (with Inception-ResNet-v2-TDM) [111] who achieves 36.8.

4 Experiments

This chapter will describe the setup and the execution of all experiments. The datasets have been are described in chapter 2.2 and the conclusions are presented in chapter 5.

The experiments were performed on all three datasets (diamond dataset, geometrical dataset, pollen dataset) mentioned in chapter 2.2.

4.1 Training Setup

For training any networks for this thesis, two similar HPC clusters were used. Subtasks of the thesis were distributed among them.

To perform the first sub-task of the thesis (classification task), the software and hardware environment of the ML partition of the TU Dresden cluster (Taurus) [78] [76] was used. This cluster is called the HPC-DA system.

With the HPC-DA system, the TU Dresden provides the infrastructure for High-Performance Computing and Data Analysis for computing projects with a focus on one of the following areas:

- Machine learning scenarios for large systems
- Evaluation of various hardware settings for large machine learning problems, including accelerator and compute node configuration and memory technologies
- Processing of large datasets on highly parallel infrastructure.

Thereby, the HPC-DA systems is a suitable platform for the thesis realisation.

HPC-DA system is built from IBM Power9 nodes [77]. HPC-DA system includes 32 IBM AC922 nodes with this configuration for each node:

- 2 x IBM Power9 CPU (2.80 GHz, 3.10 GHz boost, 22 cores)
- 256 GB RAM DDR4 2666MHz
- 6x NVIDIA VOLTA V100 with 32GB HBM2
- NVLINK bandwidth 150 GB/s between GPUs and host

To implement the used ML networks, PyTorch/1.6.0 was used.

For the classification task, the HPC-DA system was used. For the purpose of the training one node with 4 NVIDIA VOLTA V100 was used. The job was distributed by PyTorch distributed data-parallel tool.

As was written before, HPC-DA is built on the basis of Power9 architecture from IBM. The main feature of the Power9 architecture (ppc64le) is the ability to work with the NVIDIA Volta V100 GPU with NV-Link support [6]. The Power9 architecture is not as common as the x86 architecture. This means that not all applications and packages have the support of this architecture. In particular, the

TorchRay package for Pytorch, which implements the extreme perturbation function (see 3.3.2), does not support ppc64le. The Taurus partition based on the x86 architecture provides NVIDIA k20 and k80 on board. Thus, a different HPC cluster was used for other subsections of the thesis.

For the thesis sub-task number 2 (extract representation of neural network) and sub-task number 3 (localisation task) HPC Helmholz Zentrum Dresden-Rossendorf [104] cluster named Hemera was used. An important feature of the Hemera cluster is the availability of NVIDIA V100 / P100 GPU, on the x86 architecture.

Hemera contains 88 CPU nodes each with 40 Intel Xeon Gold and 24 GPU nodes each with 28 cores and 4 Nvidia GPUs type Tesla P100 or V100. The network of the hemera cluster is constructed of an EDR InfiniBand (100Gbit/s).

For the purpose of sub-task number 2 and sub-task number 3 one node with 1 NVIDIA VOLTA V100 was used. To implement the used ML networks, PyTorch/1.6.0 was used.

4.2 Classification

4.2.1 Experiment's Progress and Results

According to sub-task No. 1, it is necessary to make a classification. The course of the experiment in terms of classification will be presented. For classification, the architecture ResNet [46] was used. For the purpose of the thesis, two types of ResNet architecture was used: ResNet18, ResNet50. A comparison has been made. Details can be found in table 2.

Each dataset was split into two datasets. The first sub-dataset contains only images with only one kind of object per image. The second sub-dataset contains images that contain a combination of different types of objects. For simplicity, hereinafter in the text, the first sub-dataset (one kind of objects per image) is named "simple", and the second sub-dataset (different kinds of objects per image) is named "mixed" dataset. Thus, the geometric mixed dataset contains the categories "core", "halo", "control" in a single image (see 2.2.3). The pollen mixed dataset contains 'equal split', 'smaller pollen split', 'middle pollen split', 'bigger pollen split' in a single image (see 2.2.2).

As preparation for the training, each sub-dataset was randomly split into 3 parts: training, test and validation part(with ratio 80% 15% 5% respectively).

The training was done in 20 epoch with batch size 32 for ResNet18 and batch size 16 for ResNet50.

SGD optimizer was used with the Reduce learning rate scheduler. Cross-Entropy loss, 2-fold cross-validation are implemented.

Models were trained on the training data and tested on the 'test' data. The results are presented in Table 2. The table also shows the accuracy of the test data. The F1 score is also presented in the table. This value can be interpreted as a weighted average of precision and recall [3].

Table 2: Classification results. Comparison of results obtained for datasets for ResNet18 and ResNet50 architectures

Dataset	model architecture	Accuracy (test set)	F1	AP**	AR**
Geometric dataset simple	ResNet18	0.984	0.915	1.0	1.0
Geometric dataset mixed	ResNet18	0.710	0.812		
Geometric dataset mixed	ResNet50	0.713	0.815	0.815	0.889
Pollen dataset simple	ResNet18	1.000	1.000	1.0	1.0
Pollen dataset mixed	ResNet18	0.672	0.677		
Pollen dataset mixed	ResNet50	0.664	0.678	0.594	0.625
Pollen dataset full*	ResNet18	0.834	0.805		
Pollen dataset full*	ResNet50	0.835	0.812		
Diamond dataset	ResNet18	0.475	0.125		
Diamond dataset (pretrained)	ResNet50	0.575	0.130		

*Full dataset contains all instances of pollen dataset. ** AP - Average Precision, **AR - Average Recall (See Section 4.4.1, [2]).

The images in the dataset were pre-processed. Within its framework, augmentation and normalization were carried out.

The augmentation of Image data technique was used to artificially expand the size of a training dataset to improve the performance and ability of the model to generalize. Random horizontal mirroring and random rotation were applied to each instance of the dataset. Also, the images have been converted to grayscale pixel intensities.

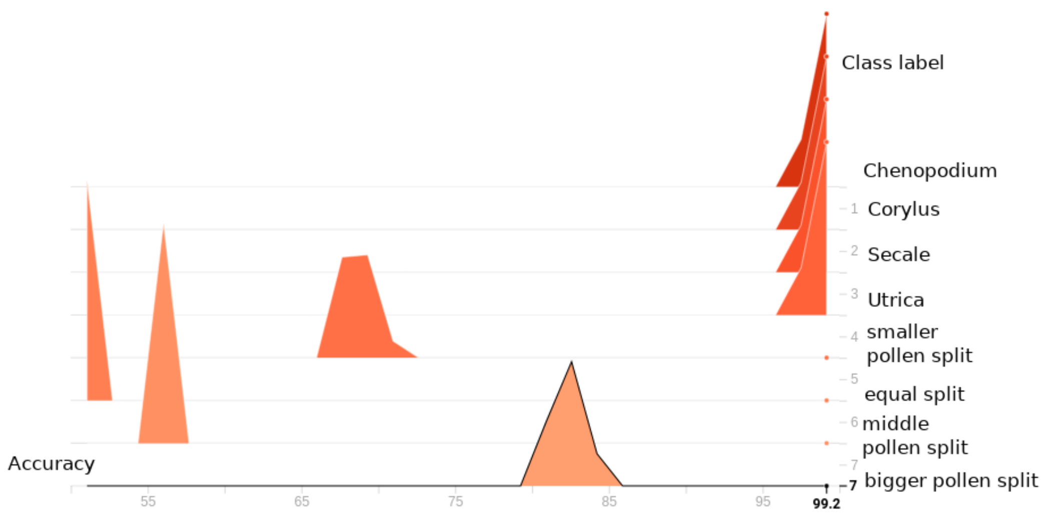

Figure 15: Accuracy for each class for full pollen dataset

Table 3: Classification accuracy results per class for sub-datasets

Name of the class	model architecture	Accuracy	Name of the class	model architecture	Accuracy
Geometric simple dataset:	ResNet18		Pollen simple dataset:	ResNet18	
Circles		0.99	Chenopodium		1.00
Triangles		1.00	Corylus		1.00
Rectangles		0.89	Secale		1.00
-	-	-	Urtica		1.00
Geometric mixed dataset:	ResNet50		Pollen mixed dataset:	ResNet50	
Core		0.86	bigger pollen split		0.81
Halo		0.18	equal split		0.51
Control		1.00	middle pollen split		0.55
-	-	-	smaller pollen split		0.69

The accuracy results for classes (categories) for each dataset are presented in the table 3.

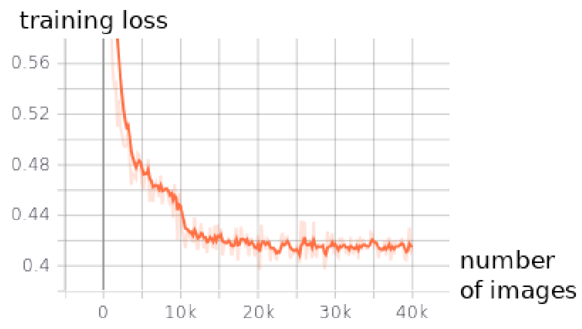

Figure 16: Relation of training loss from number of images for pollen simple dataset

4.2.2 Intermediate Summary of Classification results

Classification accuracy results per class for sub-datasets presented in Table 3. The diamond dataset shows the lowest accuracy around other datasets. To obtained better results for the diamond dataset, the ResNet50 were pretrained on a similar SEM-based dataset [82] and tested on the diamond dataset. However, the attempt to use the pretrained neural network shows an accuracy of 10% higher than the original but the threshold sill not much higher than 50 %.

ResNet18 and ResNet50 architectures were trained on all datasets. The results presented in table 2 show that ResNet18 and ResNet50 show similar results. It is possible to see that the results of ResNet18 and ResNet50 for mixed datasets (presented as an example) differ within 1 %. Training on simple sub-datasets with ResNet18 gives almost 100% results. Based on this, it was decided to continue working with ResNet18 for simple datasets and use ResNet50 for mixed datasets.

Table 3 shows that for extreme categories, where the difference between object types is strong, the classification works well (Core, Control, Bigger pollen split, Smaller pollen split). However, for categories with an intermediate position, the classification shows the probability results in lower 50% (Halo) or not much higher than 50 % (Middle pollen split, Equal split). It correlates with the F1 score (which is connected with the balance in the dataset) 0.677 for the pollen mixed dataset and 1.00 for the pollen simple dataset.

4.3 Expainable AI methods

4.3.1 Experiment's progress and Results of the Grad-CAM method in comparison with Integrated Gradients and Occlusion method

As described earlier in the 3.3.1 the Grad-CAM method is one approach to explain deep network predictions. The method was chosen in part 3.3 as one of the potential approaches for completing sub-task number 2 requiring the use of methods of "explainable AI" to extract representations that are sensitive to the matter responsible for the network's output. According to the pipeline of the approach, these representations will be used further as an input for the localisation task (see 4.4).

Grad-CAM has been used for experiments along with other methods. Several methods from attribution-based approaches have been tested. Several methods from attribution-based approaches have been tested. Compared to Grad-CAM, the results of gradient-based attribution methods and perturbation-based attribution methods were presented. In particular, the results of Grad-CAM, integrated gradients (IG) and occlusion techniques are presented in this chapter. These methods have been selected from attribution-based approaches as showing the best results.

Methods have been applied for random images with different labels from the validation part of each dataset. The methods were used for images with both correctly and incorrectly predicted labels. Grad-CAM has been implemented by the Captum package. Captum is an ”extensible library for model interpretability built by the PyTorch” [5]. Captum has a wide range of state-of-the-art algorithms. The Integrated gradient method (IG) and Occlusion method have also been implemented using Captum.

Results of Extremal perturbation function will be presented in the 4.3.2. The results obtained by different methods will be compared in the 4.3.3.

From all the results obtained, a sample for presentation was selected for each data subset. The sample contains three images (a, b, c). For clarity, all three methods were applied to each image.

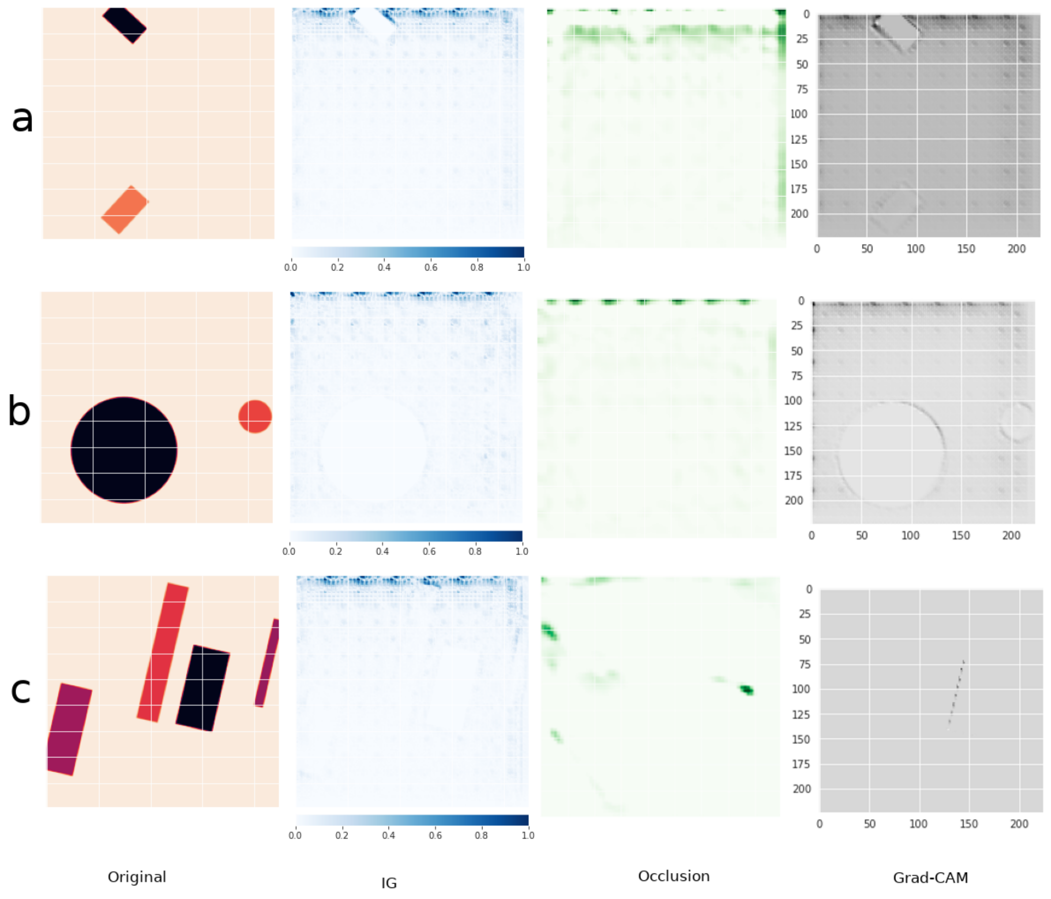

Figure 17: Images from a geometric simple sub-dataset (first column) after processing with integrated gradients, occlusion, Grad-CAM, respectively. a - true: rectangles, predicted: triangles; b - true: circles, predicted: circles; c - true: rectangles, predicted: rectangles

The results for a simple subset of the pollen dataset are presented above. Image ”a” contains rectangles and is marked accordingly. The image was mistakenly labelled as an image with triangles during classification. The labels for the images ”b” (circles) and ”c” (rectangles) were correctly predicted.

For image ”a” in figure 17, the integrated gradient method shows only one of two rectangles with many visual artefacts at the borders. The occlusion method only shows visual artefacts. After Grad-CAM processing, one rectangle can be clearly identified and the other rectangle is barely visible. Grad-CAM also shows visual artefacts. Similar results can be found for images ”b” and ”c”.

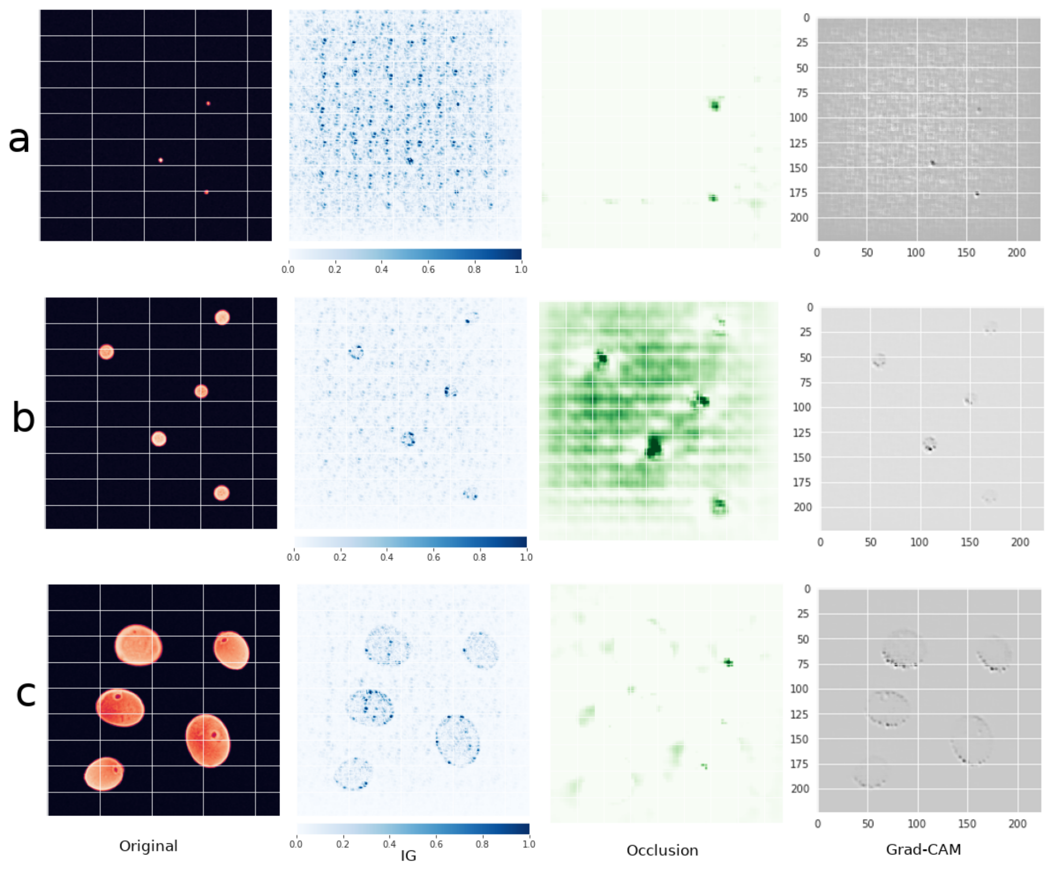

Figure 18: Images from a pollen simple sub-dataset (first column) after processing with integrated gradients, occlusion, Grad-CAM, respectively. a - true: Utica, predicted: Utica; b - true: Chenopodium, predicted: Chenopodium; c - true: Secale, predicted: Secale

The results for the 'simple' sub-dataset of the pollen dataset are presented above.

An example of results for a simple subset of pollen data is presented below. For example, image "b" in Figure 18 contains 5 Chenopodium pollen particles and has been labelled accordingly. At the time of classification, the image was correctly identified as an image with Chenopodium pollen particles. The integrated gradient method for image "b" shows all 5 particles with visual artefacts evenly distributed over the image. The occlusion method shows all 5 particles with visual artefacts. After Grad-CAM processing, 3 particles can be clearly identified and 2 particles near the boundaries are barely visible. Grad-CAM does not show visual artefacts. IG was detected false positives on the image "a". Occlusion was detected false positives for the image "c".

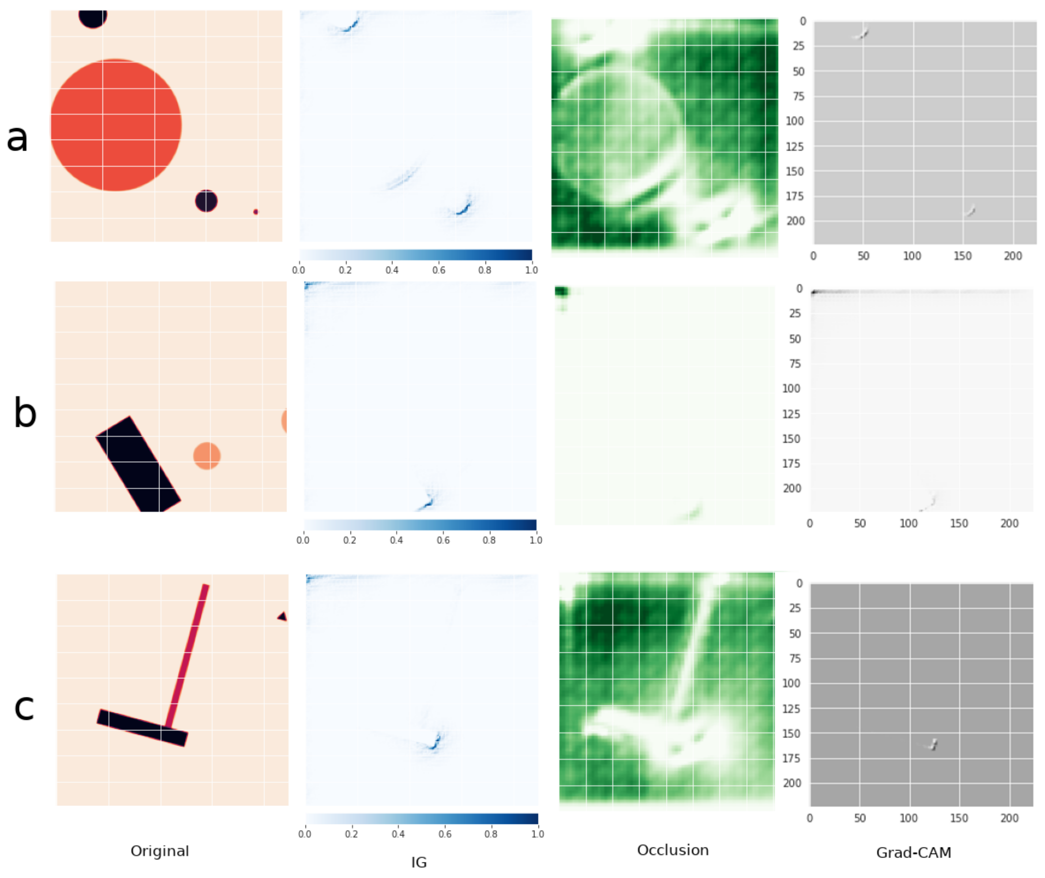

Figure 19: Images from a geometrical mixed sub-dataset (first column) after processing with integrated gradients, occlusion, Grad-CAM, respectively. a - true: core, predicted: core; b - true: halo, predicted: control; c - true: control, predicted: control

An example of results for a mixed subset of a geometric dataset is presented above. Image "a" contains three circles of different diameters and is labelled as "core". At the time of classification, the image label was correctly predicted to be "core".

The integrated gradient method shows part of the outlines of the circles. The occlusion method shows all forms, but with visual artefacts. After processing with Grad-CAM two circles can be partially identified, while other figures are not visible.

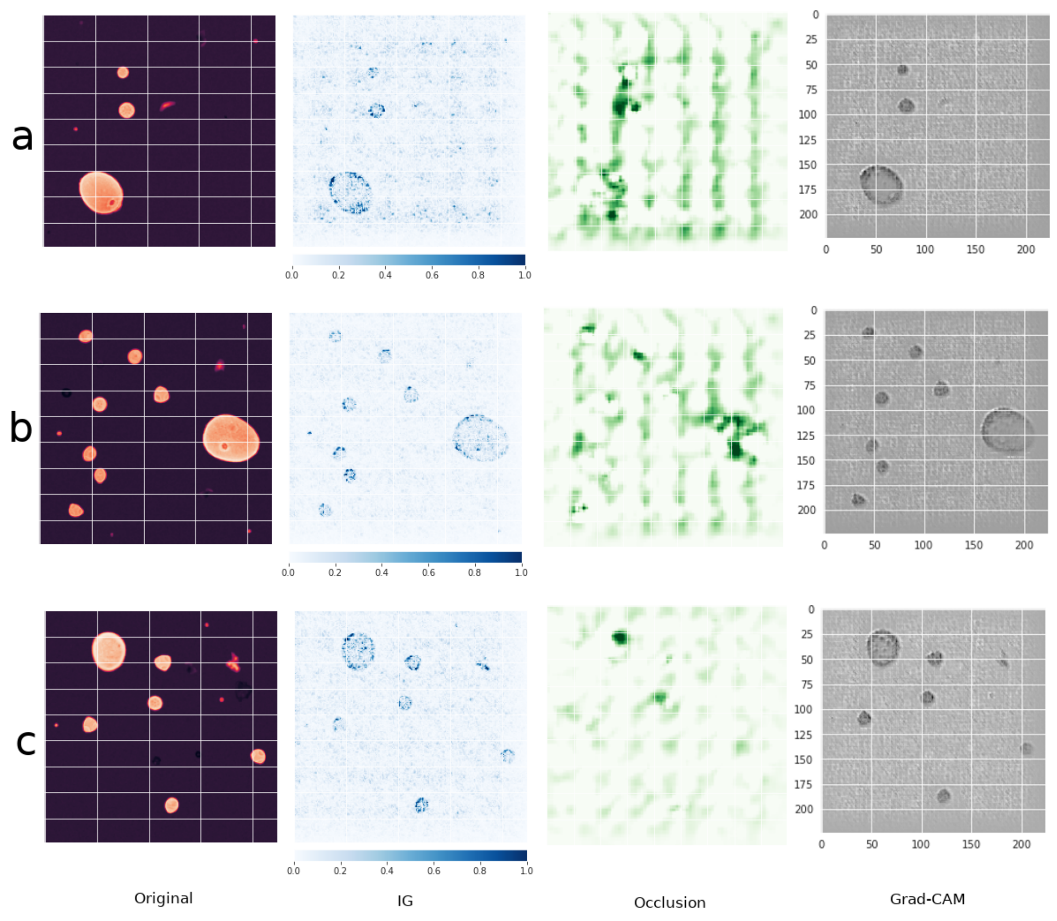

Figure 20: Images from a pollen mixed sub-dataset after processing with integrated gradients, occlusion, Grad-CAM, respectively. a - true: equal split, predicted: equal split; b - true: middle pollen split, predicted: middle pollen split; c - true: middle pollen split, predicted: middle pollen split

An example of the result of a mixed subset of pollen data is presented above. For example, image "a" in Figure 20 contains Chenopodium bonus-henricus (medium size particles), 2 Utrica dioica (smallest particles), 1 Secale cereale (biggest particle) pollen particles and one particle of foreign object. The image was marked as 'equal split'. At the time of classification, the image was correctly predicted to be an "equal split" image.

The integrated gradient method clearly shows 3 particles (1 Secale cereale and 2 Chenopodium Bonus-henricus), other particles cannot be identified behind visual artefacts evenly distributed over the image. Particles cannot be identified behind visual artefacts as a result of the occlusion method. After Grad-CAM processing 3 massive particles can be clearly identified and 2 other particles are barely visible behind visual artefacts.

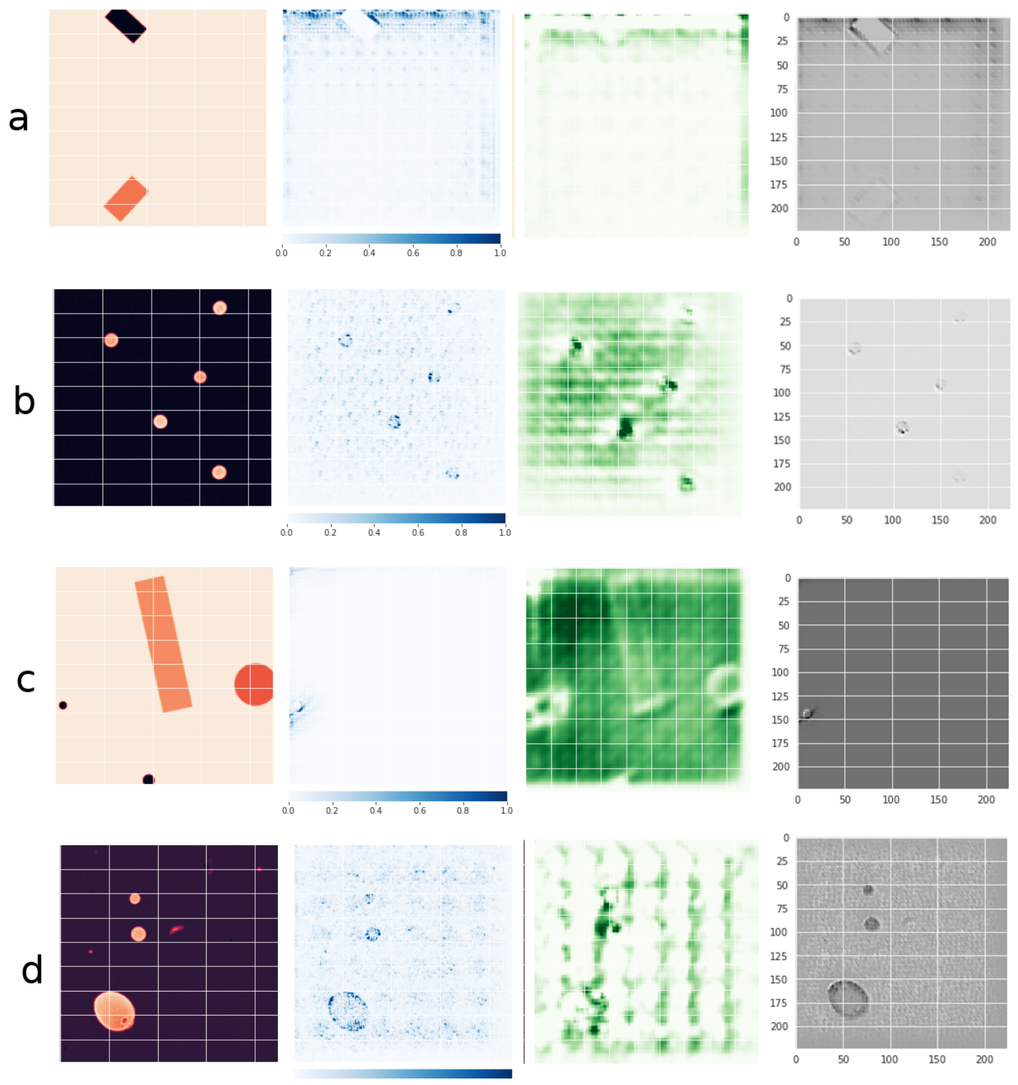

Figure 21: The sample of images from all sub-datasets after after processing with integrated gradients, occlusion, Grad-CAM, respectively. a - geometrical simple dataset; b - pollen simple dataset; c - geometrical mixed dataset; d - pollen mixed dataset

Figure 21 shows examples of images from all datasets, combined into one collage for clarity.

The following results can be found by examining the drawing 21. The occlusion method creates many artefacts in the output and rarely allows the correct image areas to be rendered. In many cases, the method cannot identify all image objects, especially for mixed datasets. At the same time, the integrated gradient method creates visual artefacts. The integrated gradient detection scheme is similar to the Grad-CAM detection scheme. Experimental results for the Grad-CAM method showed that this method can only register average network properties. In half of the cases, the method cannot identify all image objects for the mixed dataset. For a mixed pollen dataset, Grad-CAM can clearly identify massive particles, but small particles are barely visible behind visual artefacts.

4.3.2 Experiment's Progress and Results of Extremal Perturbations Method

The extremal perturbations method is a method for interpreting neural network predictions. See 3.3.2 for details. Along with the Grad-CAM method, the extremal perturbations method is one potential approach for fulfilling sub-objective number 2 of the thesis (for extracting views that are sensitive to the issue responsible for the network output). The results obtained at this stage will be used for the localisation task (see 4.4).

The pipeline for using the extremal perturbation method is the same as in the case of the Grad-Cam method: applying the method to random images with different labels from the test set of each dataset. The method was used for images with both correctly and incorrectly predicted labels. Extremal perturbations and Grad-CAM were used on the same images to compare methods and select the most appropriate approach for the thesis.

The Extremal perturbation method was implemented by Torchray [4]. The TorchRay package implements methods for visualising deep convolutional neural networks using PyTorch. The Torchray package was created by the authors of the paper [33] which originally introduced the extremal perturbation method.

The method tends to find the area of the input image that maximally excites the exact output or intermediate activation of the model. The extremal perturbation function finds a mask that increases activation.

The extremal perturbation method takes a model, an image and a target activation channel (the label). The output of the method is an optimized mask with a maximum number of activations in the channel

The extremal perturbation function allows the setup of various parameters to better tune the method for different datasets. The function allows to set up the following parameters: the parameter of the area which is a list of target areas; the number of iterations for optimizing the masks; the number of levels (blocks) with which it is possible to discretize and linearly interpolate the disturbance; mask step; mask smoothing.

Various sets of parameters have been tested. The goal was to find a set of parameters that would be stable across different images from the same dataset and universal across all datasets used in the thesis.

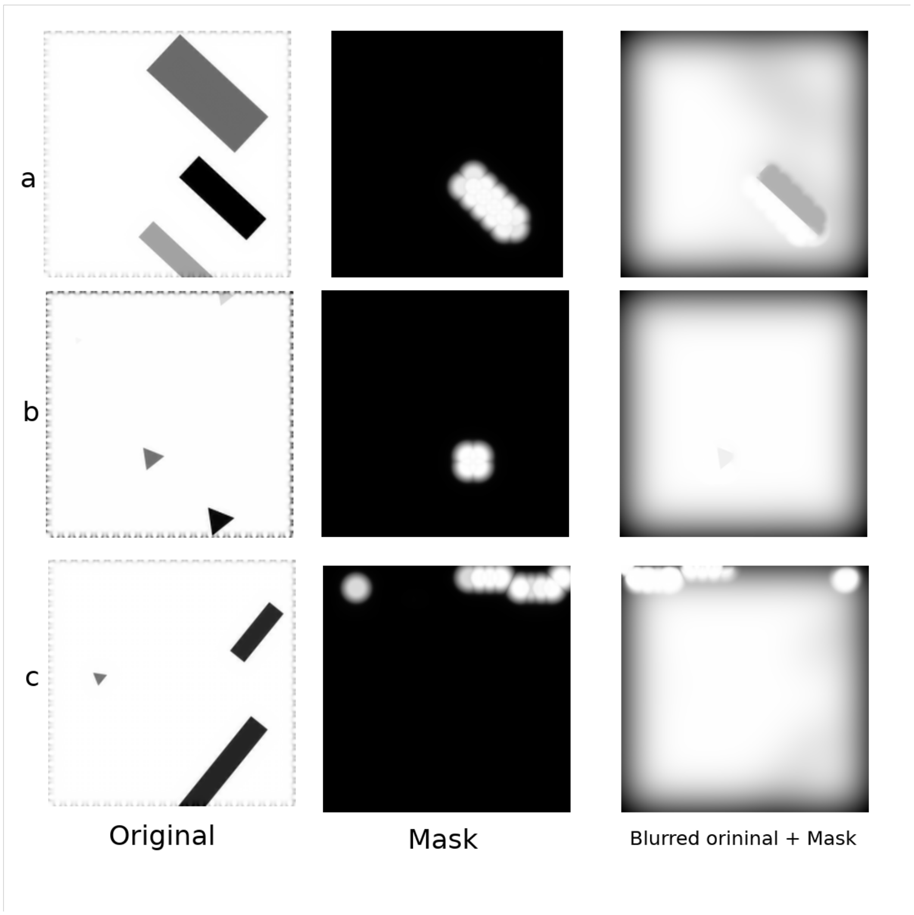

Figure 22: The image from the geometrical datasets after the Extremal perturbation method processing with set of parameters number 1. a - simple geometric dataset: true - rectangles, predicted - rectangles; b - simple geometric dataset: true - triangles, predicted - triangles; c - mixed geometric dataset: true - control, predicted - halo

The set of parameters number 1: areas [0.02]; maximum iterations 800; step/sigma 10, 12; image size [1, 1, 224, 224]; number of levels 8; mask resolution torch.Size([1, 1, 23, 23]). The image 22 shows a sample of images from a simple geometric dataset after being processed by the extremal perturbation method with the parameter set number 1. Image size reduced to (224 * 224). On a simple geometric dataset, the method shows only one shape. The method did not find any shape on the geometric mixed dataset.

The set of parameters number 2: areas [0.1]; maximum iterations 300; step/sigma 14, 12; image size [1, 1, 1280, 1280]; number of levels 10; mask resolution torch.Size([1, 1, 92, 92]). The image 45 shows a sample of images from a simple geometric dataset after being processed by the extremal perturbation method with the parameter set number 2.

On a simple geometric dataset, the method shows figures in all three examples. However, it also shows a computation artefact at the bottom of all images.

The set of parameters number 3: areas [0.1]; maximum iterations 300; step/sigma 9, 12; image size [1, 1, 1280, 1280]; number of levels 8; mask resolution torch.Size([1, 1, 143, 143]). The image 46 shows a sample of images from a simple geometric dataset after being processed by the extremal perturbation method with the parameter set number 3.

The experiment was continued with the set of parameters number 3. The extremal perturbation method with the selected set of parameters does not show artefacts and determines most of the objects in the output.

Results for all four datasets are presented. Based on the results, a sample was selected randomly for each sub-dataset. The sample contains three images (a, b, c).

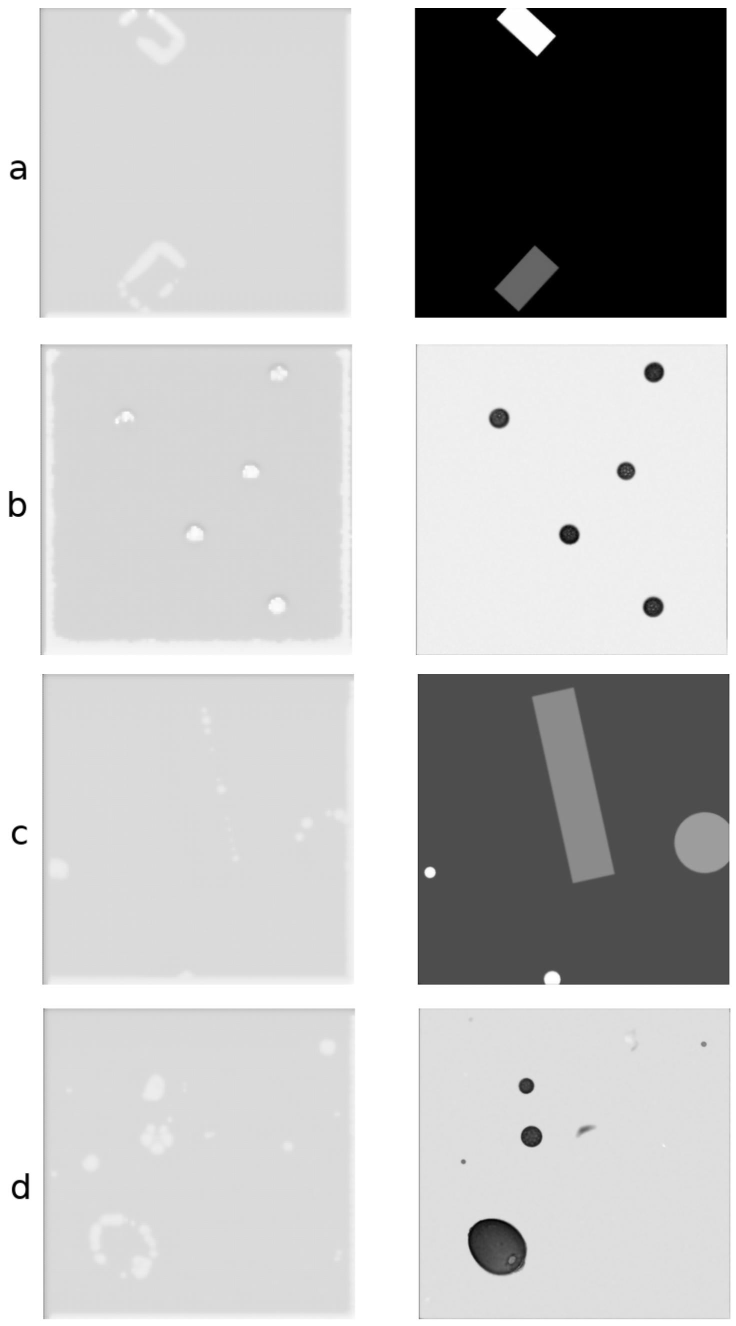

Figure 23: The sample of images from all sub-datasets after processing with the Extremal perturbation method. a - geometrical simple dataset; b - pollen simple dataset; c - geometrical mixed dataset; d - pollen mixed dataset. Left image - image after EP; Right image - original image

On a simple geometric dataset and a simple pollen dataset, the method clearly shows shapes and particles. The results of using the extremal perturbation function have a slight artefact at the boundaries. Small particles in the simple pollen dataset can be detected, but at the same time, the methods detect artefacts. On a mixed geometric dataset and a mixed pollen dataset, the method shows shapes and particles, but without clear boundaries. The method detects debris particles in the mixed pollen dataset.

4.3.3 Intermediate Summary of Explainable AI methods. Dataset for Localisation Task

To logically continue the structure of the report, it is necessary to present several brief intermediate results comparing the various methods of Explainable AI. The main method has to be chosen to complete the current sub-task and use the results of the method for the localisation task.

Thus, the method should reveal the area of the image that most likely led to the prediction of the classifier. This uncovered region is the object that has to be put a bounding box around and produce a coco-like dataset.

The main criterion for explainable AI methods was an accurate and clear understanding of the questions responsible for predicting the classifier. The selected contours or areas must match the position of objects in the original image. Visual artefacts should be kept to a minimum in order to create a bounding box for the subsequent localisation task.

The results of using explainable AI methods were presented in the previous subsections. The occlusion method shows less stable results among all methods for both simple and mixed datasets and creates a lot of visual artefacts, which makes images obtained with this method unsuitable for image processing. The differences in the results of the method are averaged, which makes the method imprecise and inappropriate for the purpose of the thesis.

The integrated gradient method shows the correct outlines of some objects, especially for simple datasets. It is more stable than the occlusion method, but creates a lot of visual artefacts, making the method imprecise and unsuitable for thesis purposes.

Grad-CAM has fewer artefacts than occlusion and integrated gradients. It shows the position of objects (their positions can be visually distinguished), but not in all cases. For mixed datasets, it can only detect few object in the image. Grad-CAM can clearly identify massive particles, but small particles are barely visible behind visual artefacts for mixed datasets.

All of these methods can capture average network properties, but cannot characterize individual outputs.

Captum was chosen as a realisation for the Integrated Gradients method, Occlusion method and the Grad-CAM method. Captum is one of the few packages that allow implementing these methods. Another important point is that the Captum package has the ability to use these methods only for images with a resolution of 224 x 224 pixels, which makes post-processing difficult.

On a simple geometric dataset and a simple pollen dataset, the extremal perturbation method clearly shows shapes and particles. It shows stable results. The results of using the extremal perturbation function have a slight artefact at the boundaries. At the same time method show stable accuracy for mixed datasets.

As a result of the comparison of the data from of presented methods given in 4.3.1, 4.3.2, the Extremal perturbation method was chosen as the main method in order to complete the sub-task number 2. The next step is to prepare a dataset from the results of the method for the localization task (sub-task number 3). To complete the final sub-task it was decided to use a coco-like dataset [69].

The extremal perturbation function was applied to each image from the validation dataset. The result of using the extremal perturbation function is a tensor that can be shown as an image. To prepare the dataset, the mask used by the extremal perturbation function was extracted as a black and white image (i.e., a binary mask). This was done for every image of a validation set of every used dataset. Since the extremal perturbation function in many cases can only capture the outline of an object or its scattered parts, it is necessary to use image processing methods to create a single mask for one object in the image. For this purpose, the morphological functions (such as closing and dilation) from scikit-image package [119] were used. Further, for each object on the image, the binary mask annotations were encoded into a png image. The pycococreator tool [1] was used then to create annotations and reorganise the data into coco-like dataset. For geometrical and pollen 'simple' datasets all objects on one image were labelled with one class. For geometrical and pollen 'mixed' datasets objects on the image were labelled by label function from "measure label" tool of the scikit-image package. This function labels connected regions of an integer array.

4.4 Localisation

4.4.1 Experiment's Progress and Results

To complete sub-task number three localisation have to be used. A dataset was prepared based on the result of using the extremal perturbation function. The technique and results of using Explainable AI methods were presented in 4.3. The main purpose of this part of the work is to localise objects on signal images of datasets presented in chapter 2.2 of the thesis.

For the purpose of the thesis, the classification task is a pretext task and the localisation task has the role of a downstream task.

For the purpose of the localisation RetinaNet [71] architecture was used. As a specific implementation of the RetinaNet architecture, the PyTorch implementation from [7] was used. As an FPN part, (see 3.4) of the RetinaNet, the ResNet-50 architecture was used.

The following datasets were used for the training during the localisation:

- Simple geometrical dataset
- Simple pollen dataset
- Mixed geometrical dataset
- Mixed pollen dataset
- Diamond dataset

All datasets used in the localisation part of the work can be divided into three types:

- EP datasets - datasets with images after the processing by extremal perturbation method with annotations obtained in the process of the work;
- GT datasets - 'ground truth' datasets with original images (without EP) with annotations obtained in the process of the work;
- GT datasets with original annotations - datasets with raw/original images (without EP) with original annotations provided by the creators of the dataset (only pollen dataset)

In preparation for training, each sub-dataset was randomly divided into 2 parts: training and validation part (with ratio 90% 10%, respectively).

EP datasets and GT datasets have a coco-like dataset structure. The metrics used for the coco dataset were used to estimate the effectiveness of the RetinaNet in the downstream task for these datasets. The 'ground truth' GT dataset was prepared for each sub-datasets used in the thesis to evaluate the model objectively.

EP datasets were prepared according to the technique described in 4.3.3. GT datasets were prepared from original datasets before the extremal perturbation was applied on images. Annotations for images for EP and GT datasets were made by the pycococreator tool [1]. To obtain the masks required for pycococreator to work, a similar technique was used as in 4.3.3. As was written earlier for 'simple' datasets all objects on one image were labelled with one class; for 'mixed' datasets objects on the image were labelled by label function from the 'measure label' tool of the scikit-image package. This function labels connected regions of an integer array.

GT datasets with original annotations have a custom format. It contains annotations in CSV files. There are two pollen sub-datasets with this format. The reason is that the authors of the original pollen dataset were prepared annotations in this way. The original geometrical dataset doesn't have it.

In this way, for the localisation task, several experiments have been carried out. The model has been trained on EP and GT version of each sub-dataset separately. For the pollen sub-datasets, the model has been additionally trained on raw/original

”ground truth” images from these datasets (GT datasets with original annotations). The results of the validation are presented as follows.

The results of localisation are presented in Table 4, Table 5, Table 6, Table 7.

The results are presented as quality metrics for results of localisation when a model was validated on different datasets. The standard coco metrics includes 10 IoU (Intersection over Union) thresholds from 0.5 to 0.95 with a step size of 0.05. The coco metrics were used in Table 4, Table 5, Table 6.

Table 4 presented the following results: results of the model trained on EP dataset and validated on EP dataset; for the comparison results from the model trained on GT dataset and validated on GT dataset have been added for each dataset (GT mark).

For the purpose to impose a high-quality constraint on the object detection interval 0.8:1.0 was created. These results are shown in Table 5, Table 6.

Table 5 presented the following results: results of the model trained on EP dataset and validated on EP dataset; for the comparison results from the model trained on GT dataset and validated on GT dataset have been added for each dataset (GT mark).

Table 6: To evaluate the model objectively, results for a model trained on EP dataset and validated on GT dataset have been presented as well. For the comparison purpose, results from the model trained on the GT dataset and validated on the GT dataset for each dataset (GT mark) have been presented as well.

Moreover, the results of the validation on GT datasets with original annotations for pollen sub-datasets have been presented in Table 7.

As a final step, localisation visualisation was performed. A sample of the results for all sub-datasets was presented in Figure 24, Figure 25, Figure 26. The objects in the output image were counted by enumerating bounding boxes.

Table 4: Quality metrics for results of localisation for different IoU thresholds in the range from 0.5 to 0.95 with a step size of 0.05 for different datasets. All models were validated on the same datasets on which they were trained.

	metrics	IoU	Max Dets	Geom. s.	Geom. s. GT	Geom. M.	Geom. M. GT	Pollen s.	Pollen s. GT	Pollen M.	Pollen M. GT	D	D GT
1	AP	all	100	0.538	0.405	0.407	0.427	0.814	0.664	0.402	0.141	0.098	0.100
2	AP	Medium	100	0.573	0.447	0.435	0.567	0.846	0.687	0.403	0.178	0.172	0.101
3	AP	large	100	0.526	0.406	0.409	0.429	0.808	0.665	0.412	0.144	0.105	0.162
4	AR	all	1	0.273	0.483	0.364	0.602	0.111	0.312	0.164	0.140	0.063	0.093
5	AR	all	10	0.746	0.750	0.670	0.780	0.727	0.962	0.635	0.686	0.387	0.227
6	AR	all	100	0.792	0.751	0.698	0.785	0.842	0.972	0.852	0.703	0.515	0.293
7	AR	medium	100	0.788	0.783	0.641	0.771	0.857	0.950	0.868	0.784	0.705	0.426
8	AR	large	100	0.800	0.742	0.719	0.784	0.846	0.982	0.823	0.839	0.338	0.300

Interpretation of terms presented in Table 4: IoU is an intersection over the union. IoU is defined as the area of intersection of the predicted bounding box and ground truth box divided by the area of the union of them. Medium objects: $32^2 <$ area (pixels) $< 96^2$. Large objects: area (pixels) $> 96^2$.

AP is an average precision, where the precision is a positive prediction value. AP can be calculated as the area under the interpolated precision-recall curve [86], [2]. AR is an average recall, where the recall is the true positive rate. AR is two times the area under the recall-IoU curve [86]. Max Dets are thresholds on max detections per image.

Designation of datasets in Table 4: Geom. s. is a Geometrical simple dataset. Geom. s. GT is a "ground truth" geometrical simple dataset. Geom. M. is a geometrical mixed dataset. Geom. M. GT is a "ground truth" geometrical mixed dataset. Pollen s. is a pollen simple dataset. Pollen s. GT is a "ground truth" pollen simple dataset. Pollen M. is a pollen mixed dataset. Pollen M. GT is a "ground truth" mixed pollen dataset. D is a diamond dataset. D GT is a "ground truth" diamond dataset.

Table 5: Quality metrics for results of localisation for different IoU thresholds in the range from 0.8 to 1.00 with a step size of 0.05 for different datasets. All models were validated on the same datasets on which they were trained.

	metrics	IoU	Max Dets	Geom. s.	Geom. s. GT	Geom. M.	Geom. M. GT	Pollen s.	Pollen s. GT	Pollen M.	Pollen M. GT	D	D GT
1	AP	All	100	0.403	0.303	0.305	0.318	0.612	0.526	0.312	0.110	0.054	0.055
2	AP	medium	100	0.443	0.348	0.316	0.439	0.647	0.544	0.311	0.138	0.107	0.041
3	AP	large	100	0.386	0.301	0.308	0.315	0.604	0.527	0.321	0.113	0.050	0.091
4	AR	all	1	0.212	0.370	0.281	0.458	0.088	0.248	0.130	0.111	0.044	0.059
5	AR	all	10	0.567	0.570	0.504	0.585	0.563	0.765	0.499	0.541	0.260	0.126
6	AR	all	100	0.592	0.570	0.521	0.588	0.634	0.773	0.664	0.554	0.335	0.148
7	AR	medium	100	0.606	0.604	0.473	0.596	0.661	0.755	0.676	0.615	0.481	0.208
8	AR	large	100	0.584	0.561	0.538	0.583	0.628	0.781	0.642	0.667	0.156	0.117

Designation of datasets in Table 5: Geom. s. is a Geometrical simple dataset. Geom. s. GT is a "ground truth" geometrical simple dataset. Geom. M. is a geometrical mixed dataset. Geom. M. GT is a "ground truth" geometrical mixed dataset. Pollen s. is a pollen simple dataset. Pollen s. GT is a "ground truth" pollen simple dataset. Pollen M. is a pollen mixed dataset. Pollen M. GT is a "ground truth" mixed pollen dataset. D is a diamond dataset. D GT is a "ground truth" diamond dataset.

Table 6: Quality metrics for results of localisation for different IoU thresholds in the range from 0.8 to 1.00 with a step size of 0.05 for different datasets with respect to images from GT datasets i.e ”ground truth” images (with annotations obtained in the process of the work).

	metrics	IoU	Max Dets	Geom. s.	Geom. s. GT	Geom. M.	Geom. M. GT	Pollen s.	Pollen s. GT	Pollen M.	Pollen M. GT
1	AP	All	100	0.177	0.303	0.175	0.318	0.164	0.526	0.057	0.110
2	AP	medium	100	0.302	0.348	0.448	0.439	0.170	0.544	0.080	0.138
3	AP	large	100	0.160	0.301	0.124	0.315	0.172	0.527	0.053	0.113
4	AR	all	1	0.208	0.370	0.249	0.458	0.148	0.248	0.036	0.111
5	AR	all	10	0.309	0.570	0.325	0.585	0.381	0.765	0.157	0.541
6	AR	all	100	0.314	0.570	0.330	0.588	0.381	0.773	0.167	0.554
7	AR	medium	100	0.398	0.604	0.626	0.596	0.348	0.755	0.202	0.615
8	AR	large	100	0.293	0.561	0.250	0.583	0.400	0.781	0.162	0.667

Designation of datasets in Table 6: Geom. s. is a Geometrical simple dataset. Geom. s. GT is a ”ground truth” geometrical simple dataset. Geom. M. is a geometrical mixed dataset. Geom. M. GT is a ”ground truth” geometrical mixed dataset. Pollen s. is a pollen simple dataset. Pollen s. GT is a ”ground truth” pollen simple dataset. Pollen M. is a pollen mixed dataset. Pollen M. GT is a ”ground truth” mixed pollen dataset. D is a diamond dataset. D GT is a ”ground truth” diamond dataset.

Table 7: Results of the Average Precision metric (IoU threshold 0.8-1.00) obtained with respect to GT datasets with original annotations (i.e raw/original "ground truth" images from pollen sub-datasets with original annotations) for the model trained: GT(o) - on raw/original "ground truth" images from these datasets; EP - on images (masks) from EP pollen sub-datasets

Name of the class	model architecture	AP	Name of the class	model architecture	AP
Pollen simple dataset:	GT(o)		Pollen simple dataset:	EP	
Chenopodium		0.156	Chenopodium		0.041
Corylus		0.131	Corylus		0.00
Secale		0.203	Secale		0.00
Urtica		0.0	Urtica		0.00
Pollen mixed dataset:	GT(o)		Pollen mixed dataset:	EP	
Chenopodium		0.088	Chenopodium		0.0136
Corylus		0.084	Corylus		0.011
Secale		0.181	Secale		0.108
Urtica		0.0	Urtica		0.0

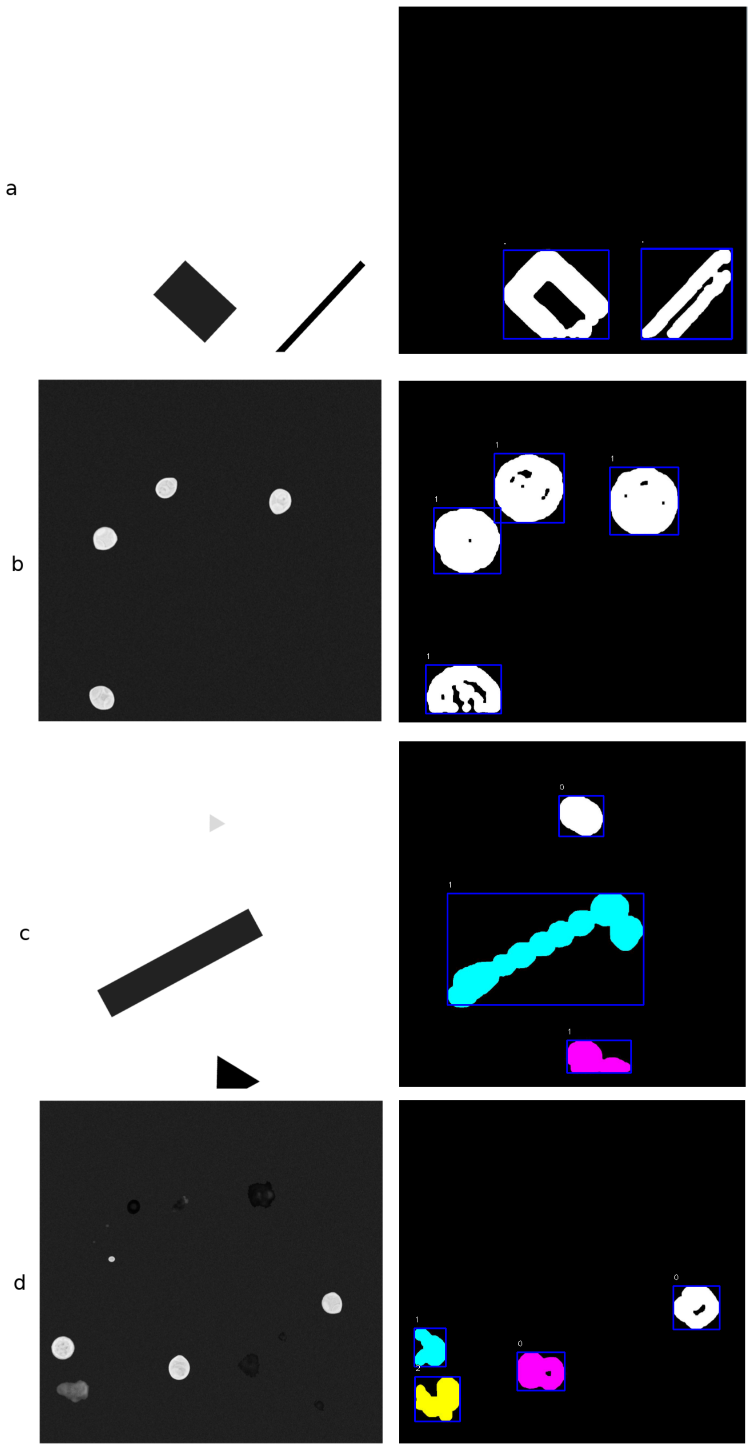

Figure 24: The sample of images from **EP datasets** (i.e images after EP from all sub-datasets with annotations obtained in the process of the work) after applying the studied approach where the model was trained on EP datasets (masks). a - geometrical simple dataset; b - pollen simple dataset; c - geometrical mixed dataset; d - pollen mixed dataset. Left image - an original image; Right image - an image after using the method

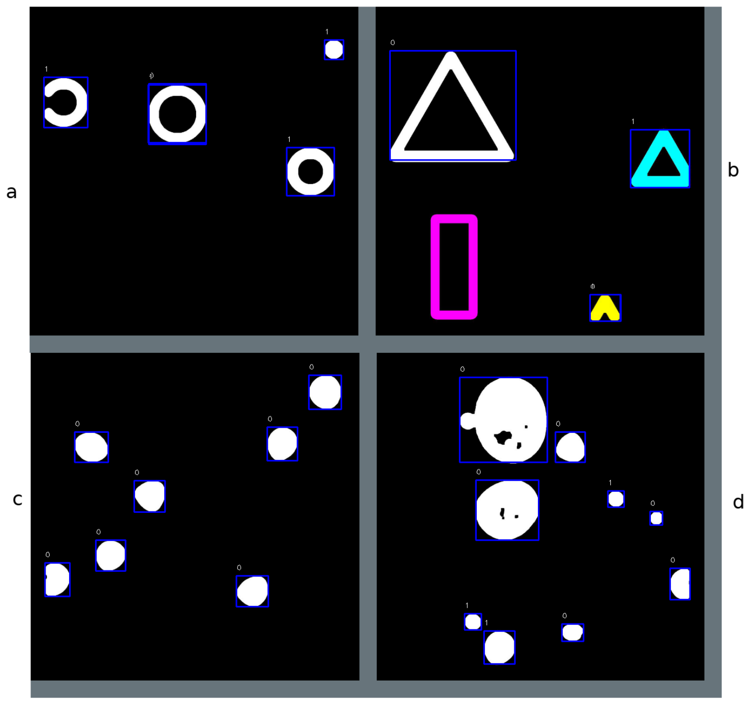

Figure 25: The sample of images from **GT datasets** (i.e "ground truth" images from all sub-datasets with annotations obtained in the process of the work) after applying the studied approach where the model was trained on EP datasets (masks). a - geometrical simple dataset; b - geometrical mixed dataset; c - pollen simple dataset; d - pollen mixed dataset.

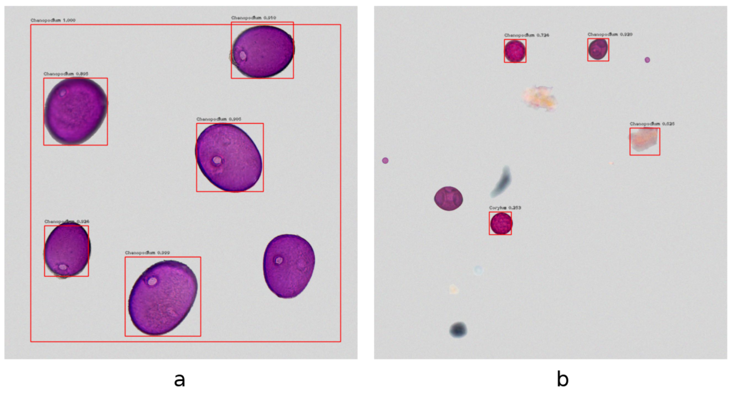

Figure 26: The sample of images from **GT datasets with original annotations** (i.e raw/original "ground truth" images from pollen sub-datasets with original annotations) after applying the studied approach where the model was trained on EP images (masks). a - pollen simple dataset; b - pollen mixed dataset.

4.4.2 Intermediate Summary of Localisation Results

To analyze the results of the work, the following should be recalled. To obtain the results in presented tables with quality metrics the classification was made and the EP was run on the results of classification. Once EP "converged", labels were assigned to all regions that EP uncovered for each class. The next step was localisation. The flow of the thesis pipeline is presented in Figure 27.

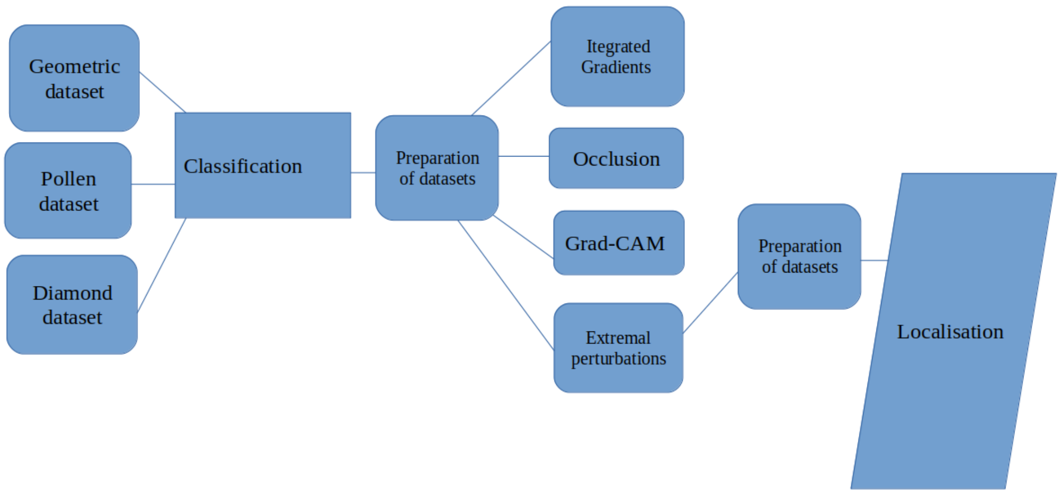

Figure 27: The pipeline of the thesis

The primary metrics for the Tables 4, 5, 6 are row number 1 and row number 6.

The models trained on the GT dataset and also validated on the GT dataset show general structure and information about the datasets themselves (Table 5). The model trained on EP dataset and validated on EP dataset shows the effectiveness of the model (RetinaNet) on the localisation task (Table 5).

The model trained on EP dataset and validated on GT dataset show the results of using the studied approach in general regarding selected sub-dataset. In this way Table, 6 shows the main results of the thesis.

The best results were obtained with a simple pollen dataset and a simple geometric dataset(see Table 6). These results are expected for simple datasets because images with only one object type per image are the simplest material for the model. See the Figure 24, Figure 25. However, the geometric mixed dataset shows result almost on the same level as a simple geometric dataset. At the same time, the comparison of the results obtained for the simple pollen dataset for the model trained on the EP and GT datasets and validated on the GT dataset shows a bigger gap than similar results for the simple geometric dataset and geometric mixed dataset.

The best result for a simple pollen dataset obtained for the model trained on the GT dataset and validated also on the GT dataset can be attributed to the homogeneity of one class of a simple pollen dataset. Each dataset class differs from others in both shape and size. Whereas in a geometric simple dataset, the sizes of figures of the same class can be different, See Figure 54.

For the pollen dataset, the precision drops from simple to mixed substantially (comparing both for datasets after EP and GT datasets); That indicates that RetinaNet mixes up different classes of pollen quite easily. See Figure 25, Figure 24, Figure 62. However, it must be said that visual analysis of post-EP images from mixed datasets prepared for the localization problem shows that the labels assigned by the algorithm rarely correspond to real "ground truths" labels. At the same time data from Table 7 shows that there is a difference in results for GT dataset with original annotations between the pollen simple and pollen mixed datasets.

Both precision (AP) and recall (AR) show the values where the geometric simple GT dataset is on par with the values of the geometric mixed GT dataset (see Table 6). However, one might expect that the task of detecting objects from several classes is more difficult. The reason may be that the dimensions of the geometric shapes for the geometric dataset with the same label can be different. This can be difficult for the model (See Figure54).

Originally, the coco-like metrics has a "small" scale for the IoU. Objects in the coco dataset that define an area less than 32x32 belong to small. However, the tables above don't have this scale. The reason for that the RetinaNet with the proposed approach can not localize small objects. However, for the geometrical simple dataset, the values 0.075 for AR and 0.077 for AP were found for the "small" scale. For the pollen simple dataset, the value was 0.01 for both AP and AR. Probably, one of the potential explanations is that the size of objects in images is bigger than 32*32. For the "Utrica" category (smaller particles) in the pollen simple dataset the size of one particle on the edge of this value. According to Table 7 the value of AP is 0.0 even for the model trained on the GT dataset with original annotations and validated on this dataset.

In general, comparison of results for AP obtained for the models trained on GT datasets and validated also on the GT datasets presented in Table 6 and in Table 7 show that the approach (for annotating datasets without original annotations) used in the thesis can be applied for the preparation of datasets for the localisation.

At the same time model trained on the EP dataset and validated on the GT datasets with original annotations (see Table 7) shows worse results than models trained on GT datasets with original annotations and validated also on this dataset. It shows the necessity of prepossessing images for validation of models trained on EP datasets.

The results presented in the Table 5 generally correlate with the results presented in the Table 4.

The value of AR in the Table 5 has a average value higher than 55% and in the Table 4 higher than 70%. It indicates the general suitability RetinaNet architecture for the presented datasets.

The Diamond dataset presented in the tables 5, 4 performed poorly. The reason for that probably that the size of the dataset not enough for the approach. The localisation result even for GT dataset AP = 0.055 correlates with a low accuracy result (0.575) for classification.

AP and AR result in Table 6 show similar values for models trained on EP datasets and GT datasets for medium IoU area for geometric simple and geometric mixed datasets . In this way, RetinaNet performs best on the middle objects of the geometric dataset.

Localisation visualisation was performed. A sample of the results for all sub-datasets was presented in Figure 24, Figure 25, Figure 26. The results for all datasets separately presented in Appendix D.

Results of the visualisation are correlates with results in tables 6, 4. For the geometric simple and mixed datasets, the number of counted objects in many cases correlates with the real number of objects in the images (counted by rough visual estimation).

The average difference for all datasets between the results obtained for the model trained on the EP dataset validated on the GT dataset, and the model trained on the GT dataset validated on the GT dataset is roughly 50%. At the same time, AP and AR result for the medium size objects of geometric simple and geometric mixed datasets (see Table 6) shows that RetinaNet trained on the EP datasets performs on GT datasets almost as good as RetinaNet trained on "ground truth" GT datasets.

5 Conclusion and Outlook

In this chapter, the results of this work are summarized. Finally, open points of the work are addressed that allow further investigations and improvements for the presented approach.

Deep neural networks with a combination of self-supervised learning and explainable AI have been used to localize objects in images from a scanning electron microscope (SEM) datasets (or similar datasets). To accomplish the task, HZDR and TU Dresden HPC clusters were used. The experimental setup and infrastructure are described in 4.1, and the experiment progress, technical details and problems are described in 4.2, 4.3, 4.4.

5.1 Conclusion

The main objects of research were geometric and pollen datasets. The classification was done for all datasets. The results obtained during the classification show the validity of the choice of ResNet architecture. The accuracy for simple datasets approaches 100 % with F1 0.915 - 1.000 for ResNet18. Mixed datasets show an accuracy of 67% - 71% for both ResNet18 and Resnet50. These indicators are at an adequate level, and this shows the possibility of using a ResNet-based model for SEM datasets and use these models as a pretext task for self-supervised learning.

The experiment carried out and the review of articles showed the difference between the approaches of various methods of Explainable AI, the results of these methods, their problems and features were also considered. The main comparison was between the Grad-CAM method and the extremal perturbation method.

The integrated gradient method and the occlusion method were applied for comparison with the Grad-CAM method. The occlusion method creates a lot of artefacts in the output and rarely allows to visualise the correct areas in the image. Integrated gradients show a low accuracy, in many cases, the method cannot identify all image objects, especially for mixed datasets. At the same time, the integrated gradient method creates visual artefacts. The detection scheme of the Integrated gradients method is similar to that of the Grad-CAM method. However, all of these factors make the integrated gradient method and the occlusion method unsuitable for the purpose of the thesis.

The results of the experiments for the Grad-CAM method showed that this method can only record the average properties of the network. In half of the cases, the method cannot identify all image objects for the mixed dataset. For a mixed pollen dataset, Grad-CAM can clearly identify massive particles, but small particles are barely visible behind visual artefacts. In case the model made an incorrect prediction, Grad-CAM will not be able to clearly identify the objects in the image. Captum's implementation of the Grad-CAM method, the integrated gradients method and the Occlusion method made the use of these methods inconvenient. All of the above factors make the use of the Grad-CAM method unacceptable for the thesis.

The best results were obtained using the method of extremal perturbations. The extremal perturbations function can identify most objects in various datasets. The method works especially well with simple datasets. For mixed datasets, it can also visualise correct areas of the images, but with less precision. The method requires customization for a specific set of data. It is difficult to customize the method for different datasets. The method creates artefacts at the borders, but they can be cut off. Among the methods considered, the method of extremal perturbations is of greatest interest for the thesis problem. Based on the results of the method of extremal perturbations, a dataset was created for the localization task. The details and the method of creating the dataset were presented.

The results presented in Table 6 correspond to the performance of the localisation task for the RetinaNet trained on different datasets with respect to "ground truth" datasets. The average difference between the results obtained for the model trained on the EP dataset validated on the GT dataset, and the model trained on the GT dataset validated on the GT dataset is roughly 50%. However, AP and AR result for the medium size objects of geometric simple and geometric mixed datasets (see Table 6) shows that RetinaNet trained on the EP datasets performs on GT datasets almost as good as RetinaNet trained on "ground truth" GT datasets.

For a pollen dataset, accuracy drops significantly from a simple dataset to a mixed dataset (for GT datasets and for EP datasets). This result shows that RetinaNet mixes different classes of pollen datasets. However, it is necessary to mention that labels assigned by the algorithm during the preparation of the datasets for the localisation task were assigned not accurately. The main priority was the study of using the approach for datasets with one label per image. Using the approach to localize objects for datasets with more than one object type per image requires further study.

For a simple geometric dataset, a detailed analysis shows that objects occupying more than 30% of images were detected as several contours and were counted as several objects (see Figure 54). The same problem was found for a geometric mixed dataset. This issue was not found for the mixed pollen dataset. Additionally, in the pollen sub-datasets, Utrica dioica particles (the smallest fraction) were poorly detected due to their size for all datasets and all models (see Table 7). The extremal perturbation method found them with artefacts or did not find them at all. RetinaNet can not find them on GT datasets with original annotations.

Localisation visualisation was performed. Results of the visualisation are correlates with results in the tables 6, 4. The best results were obtained for geometrical datasets. The number of counted objects in many cases correlates with the real number of objects in the images.

The obtained data and conclusions allow speaking with restrained optimism about the moderate success of the method on simple datasets and the potential use of this method for similar datasets. The most suitable datasets for this method are simple SEM-based datasets with one type of medium-sized objects per image.

The Approach to Self-Supervised Object Localisation through Deep Learning based classification has been developed. It has been shown that the studied approach can be effective with some limitations on simple datasets with one object type per image. The result was presented for both geometrical and pollen datasets.

5.2 Outlook

The investigated approach can be effective with some limitations on simple datasets with one object type per image. However, using a deep learning classification approach to localize objects for datasets with more than one object type per image requires further study.

The setting and selection of the values of the extremal perturbation function are for further study. With a deeper understanding of how settings of extremal perturbation affect specific datasets, it possible to achieve significantly better performance. This is especially important for small objects. The study has shown that this approach cannot detect small particles or may not detect enough of them, although it is able to classify images with small particles.

In addition, learning and applying more advanced image processing techniques to prepare a post-EP dataset for a localization task can improve localization performance.

Appendices

A Results of Classification

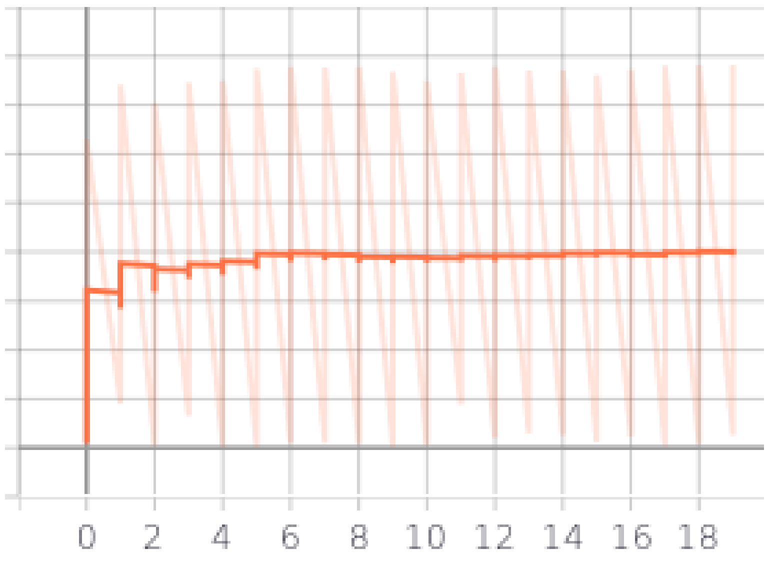

Figure 28: The diagram of correlation accuracy/epoch for full pollen dataset

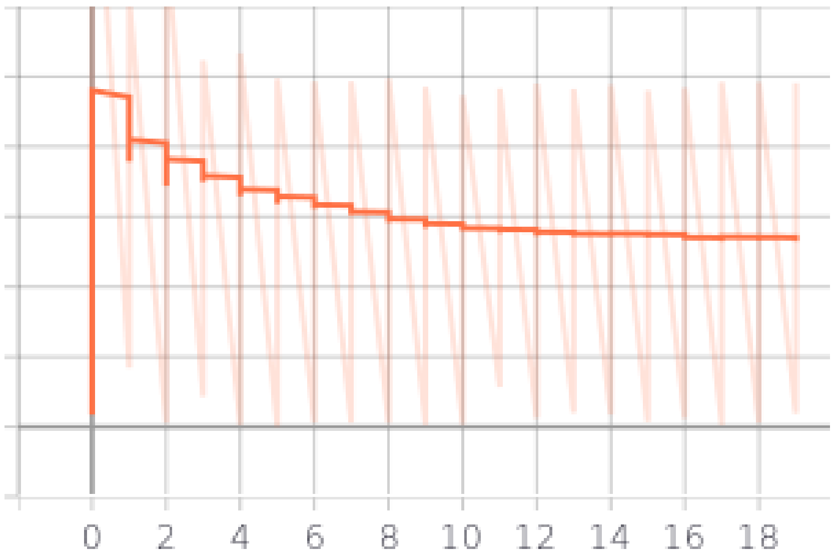

Figure 29: The diagram of correlation loss/epoch for full pollen dataset

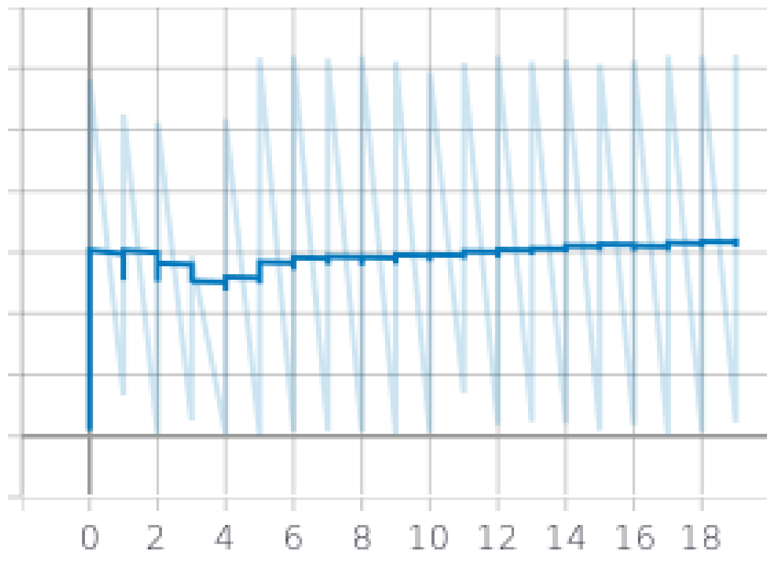

Figure 30: The diagram of correlation accuracy/epoch for mixe pollen dataset

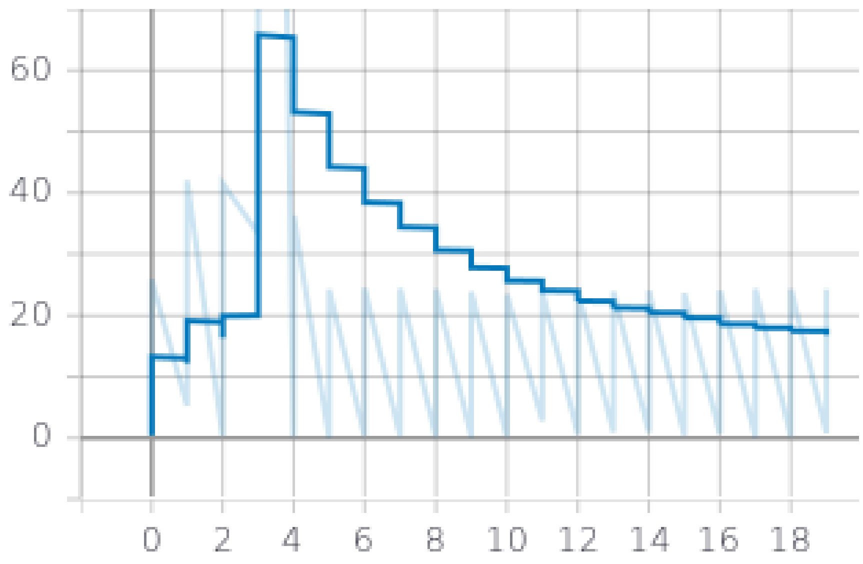

Figure 31: The diagram of correlation loss/epoch for mixed pollen dataset

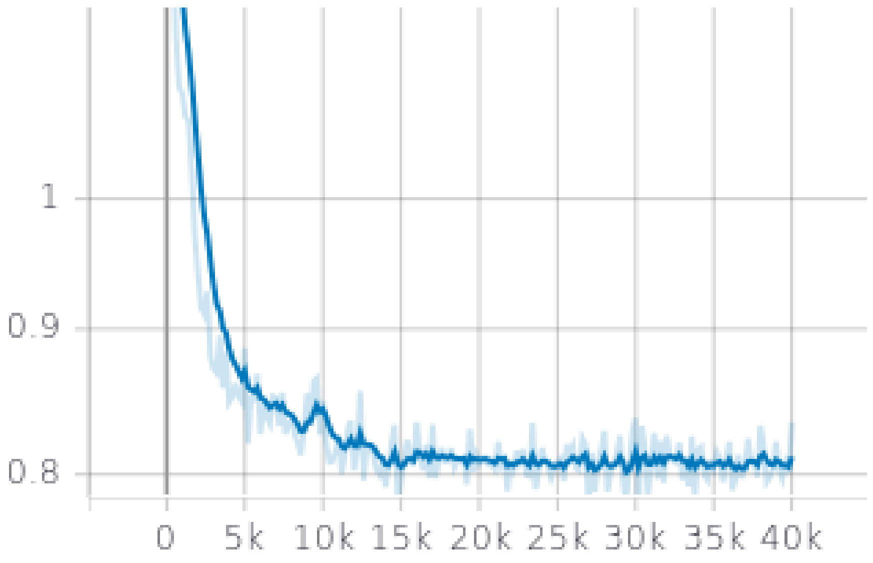

Figure 32: The diagram of correlation loss/number of images for mixed pollen dataset

B Results of Results of Grad-CAM, Integrated Gradients and Occlusion method

The following subsection shows images not listed in the main part of the thesis of listed before in lower quality.

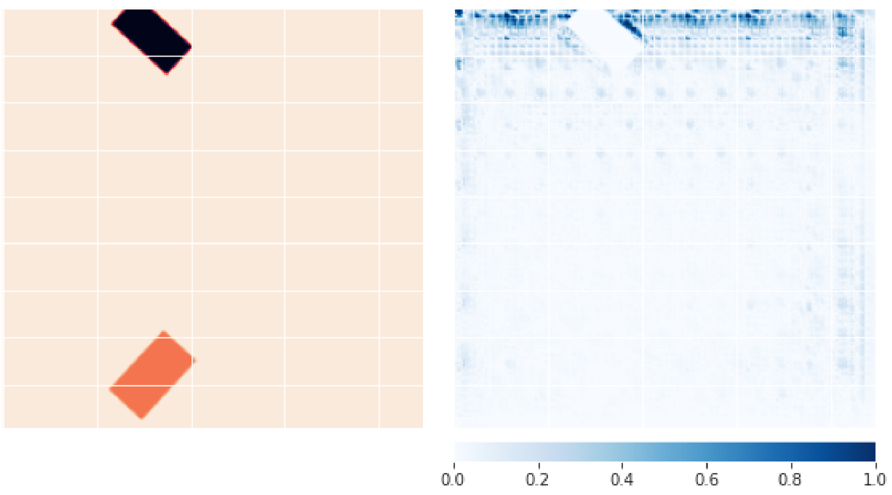

Figure 33: The image from geometrical 'simple' sub-dataset after the Integrated-gradients method processing

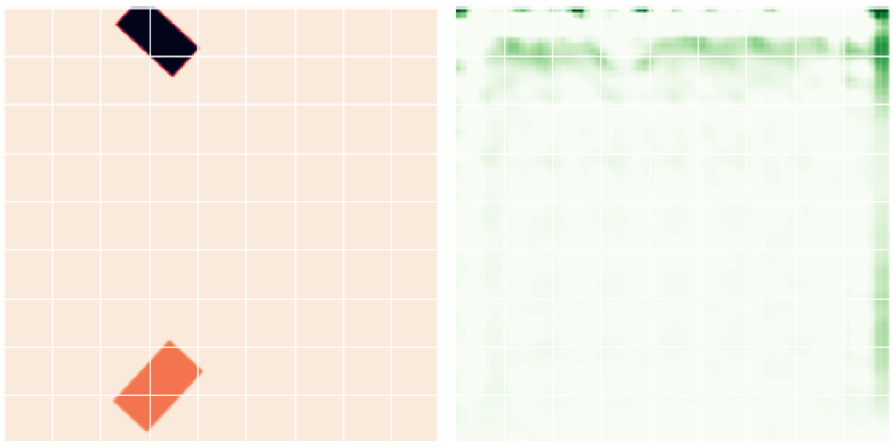

Figure 34: The image from geometrical 'simple' sub-dataset after the Occlusion method processing

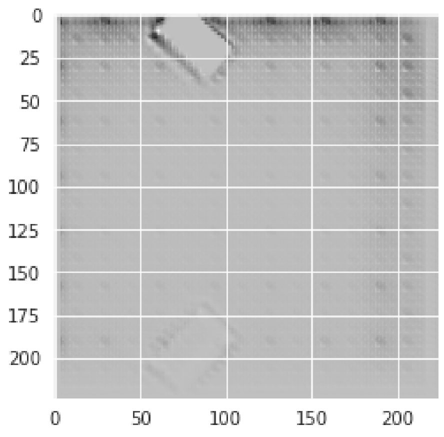

Figure 35: The image from the geometrical 'simple' sub-dataset after the Grad-CAM processing

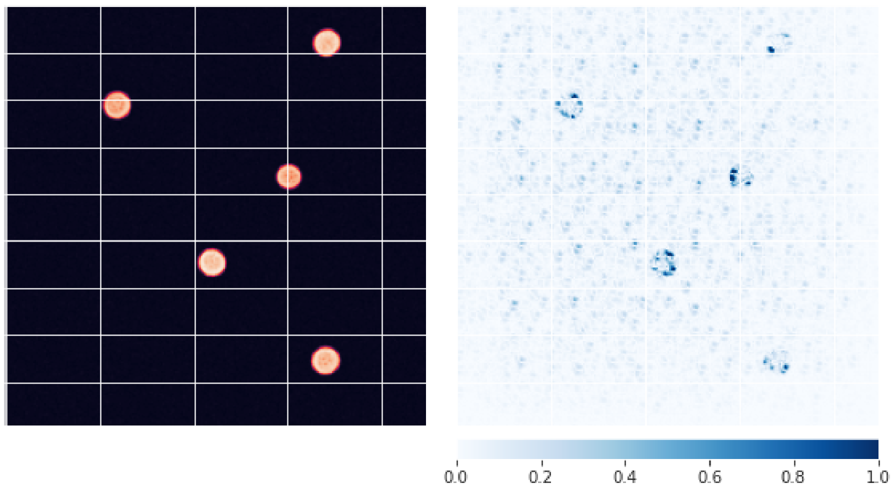

Figure 36: The image from pollen dataset after the Integrated-gradient method processing

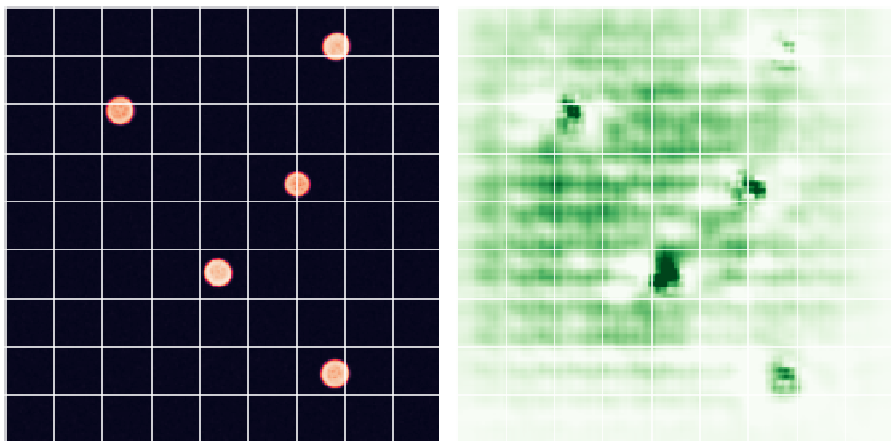
Figure 37: The image from pollen dataset after the Occlusion-based method processing

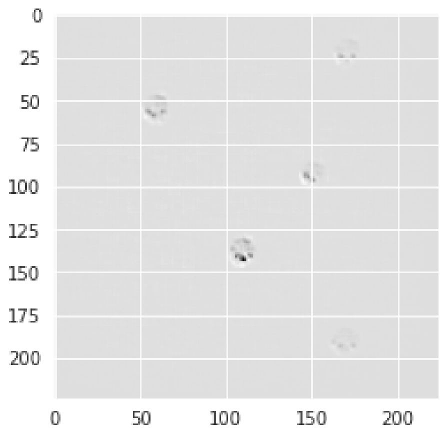

Figure 38: The image from the pollen dataset after the Grad-CAM processing

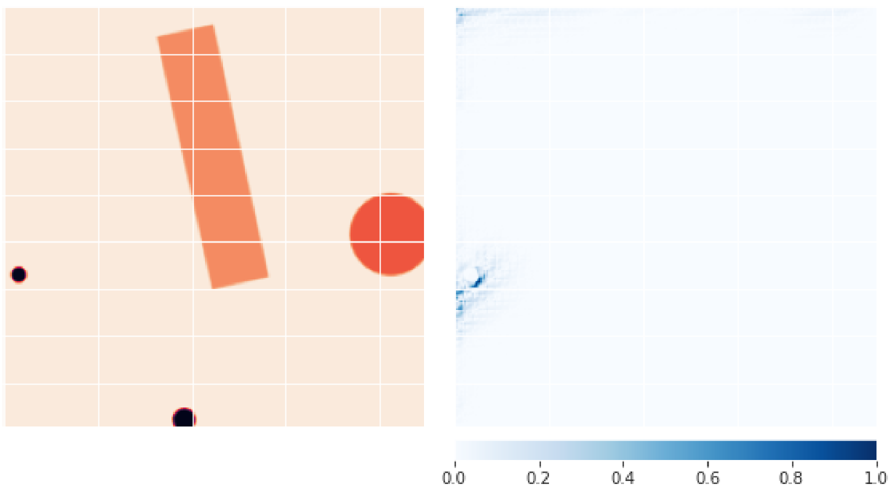

Figure 39: The image from geometrical mixed dataset after the Integrated-gradient method processing

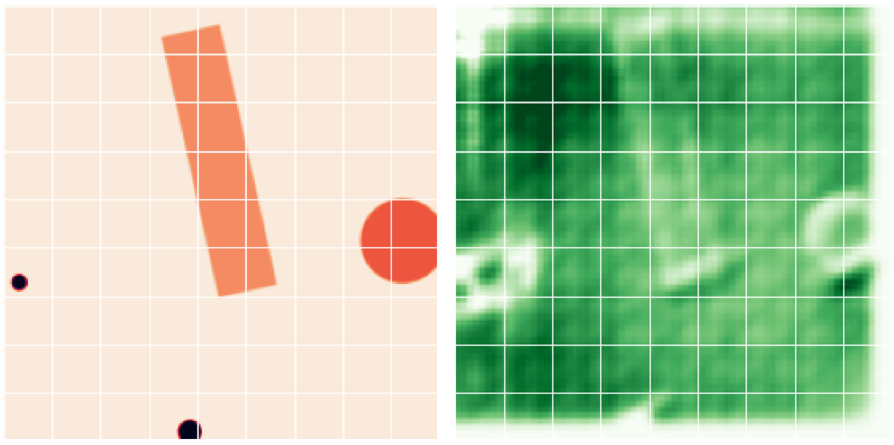

Figure 40: The image from geometrical mixed dataset after the Occlusion-based method processing

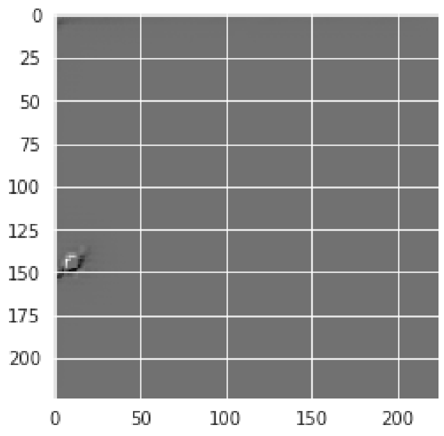

Figure 41: The image from the geometrical mixed dataset after the Grad-CAM processing

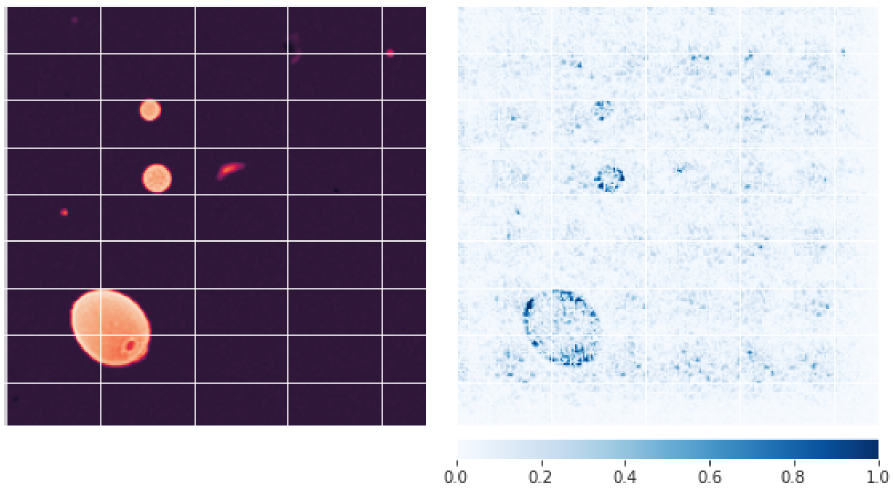

Figure 42: The image from the pollen mixed dataset after the Integrated-gradient method processing

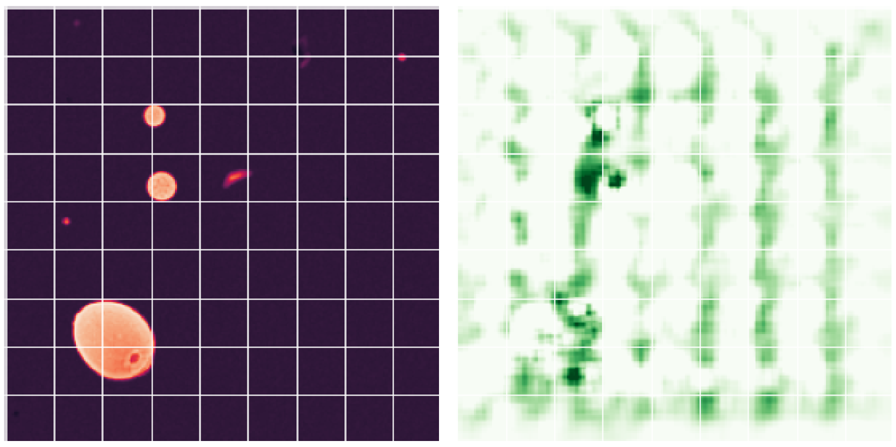

Figure 43: The image from geometrical mixed dataset after the Occlusion-based method processing

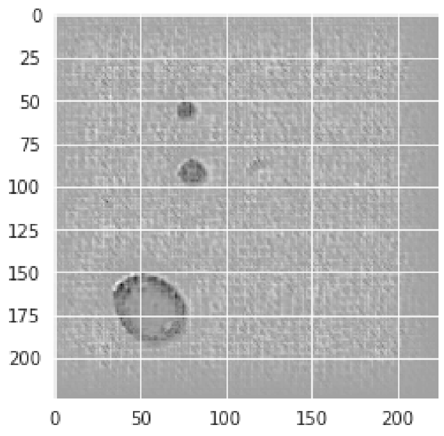

Figure 44: The image from the pollen mixed dataset after the Grad-CAM processing

C Results of Extremal Perturbations

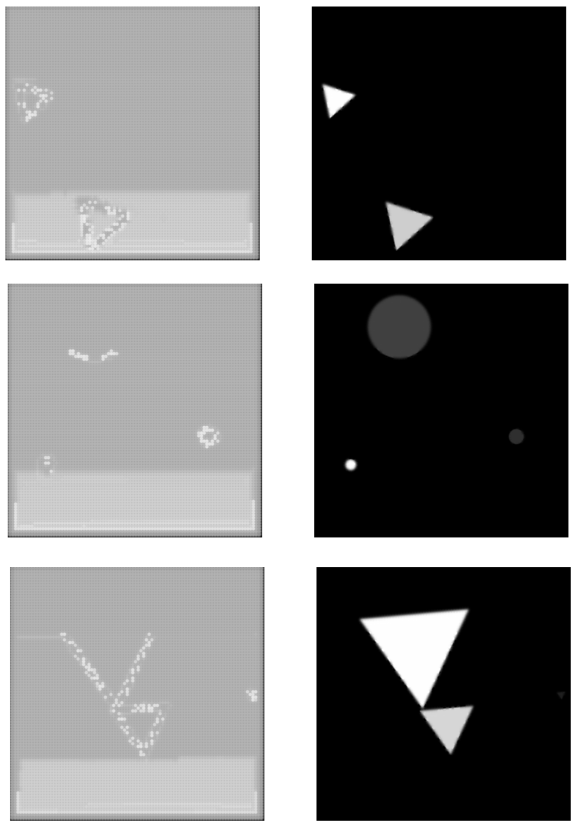

Figure 45: The image from the geometrical simple dataset after the Extremal perturbation method processing with set of parameters number 2

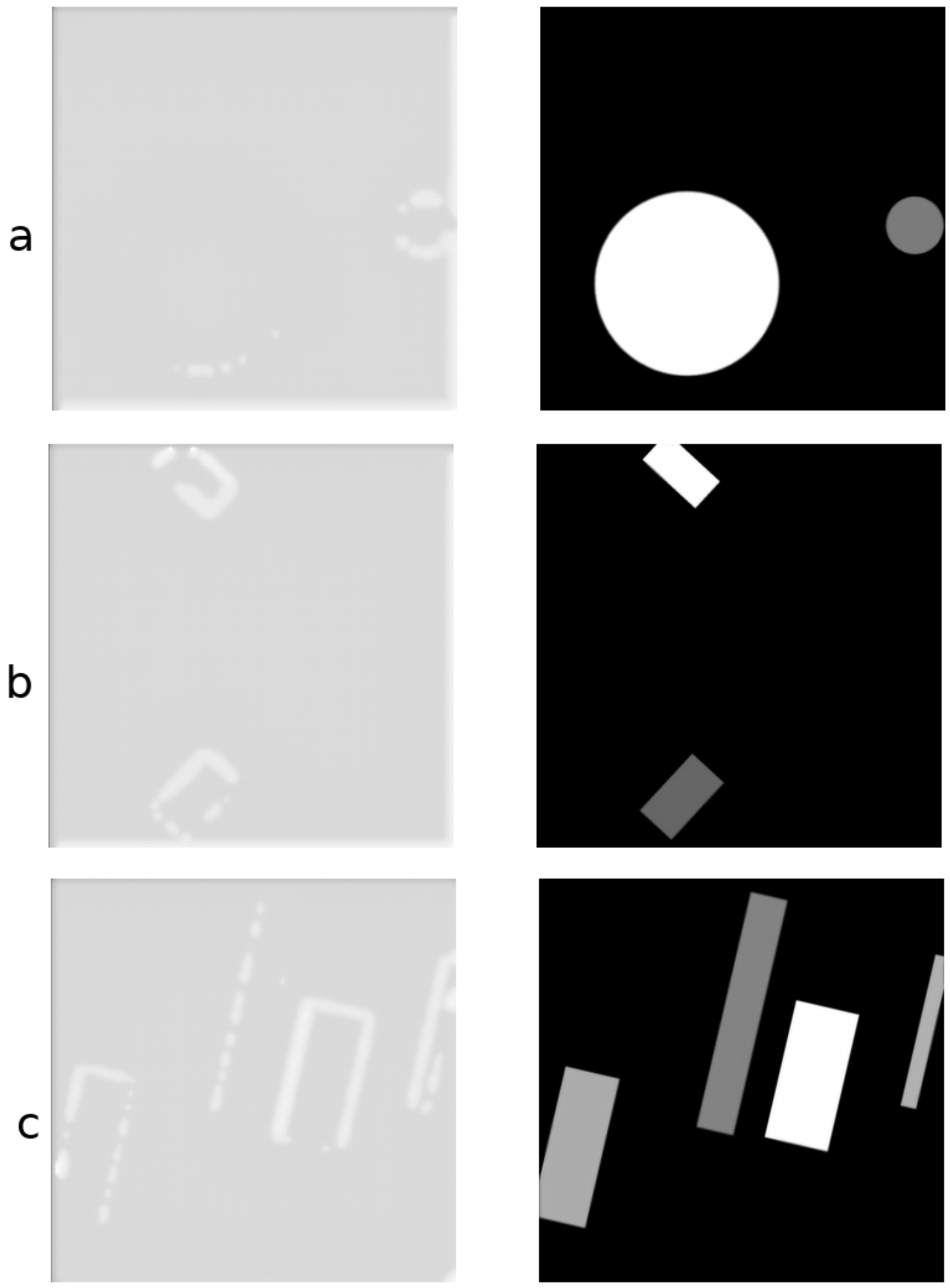

Figure 46: The image from the geometrical simple dataset after the Extremal Perturbation method processing with set of parameters number 3. Left image - image after EP; Right image - original image

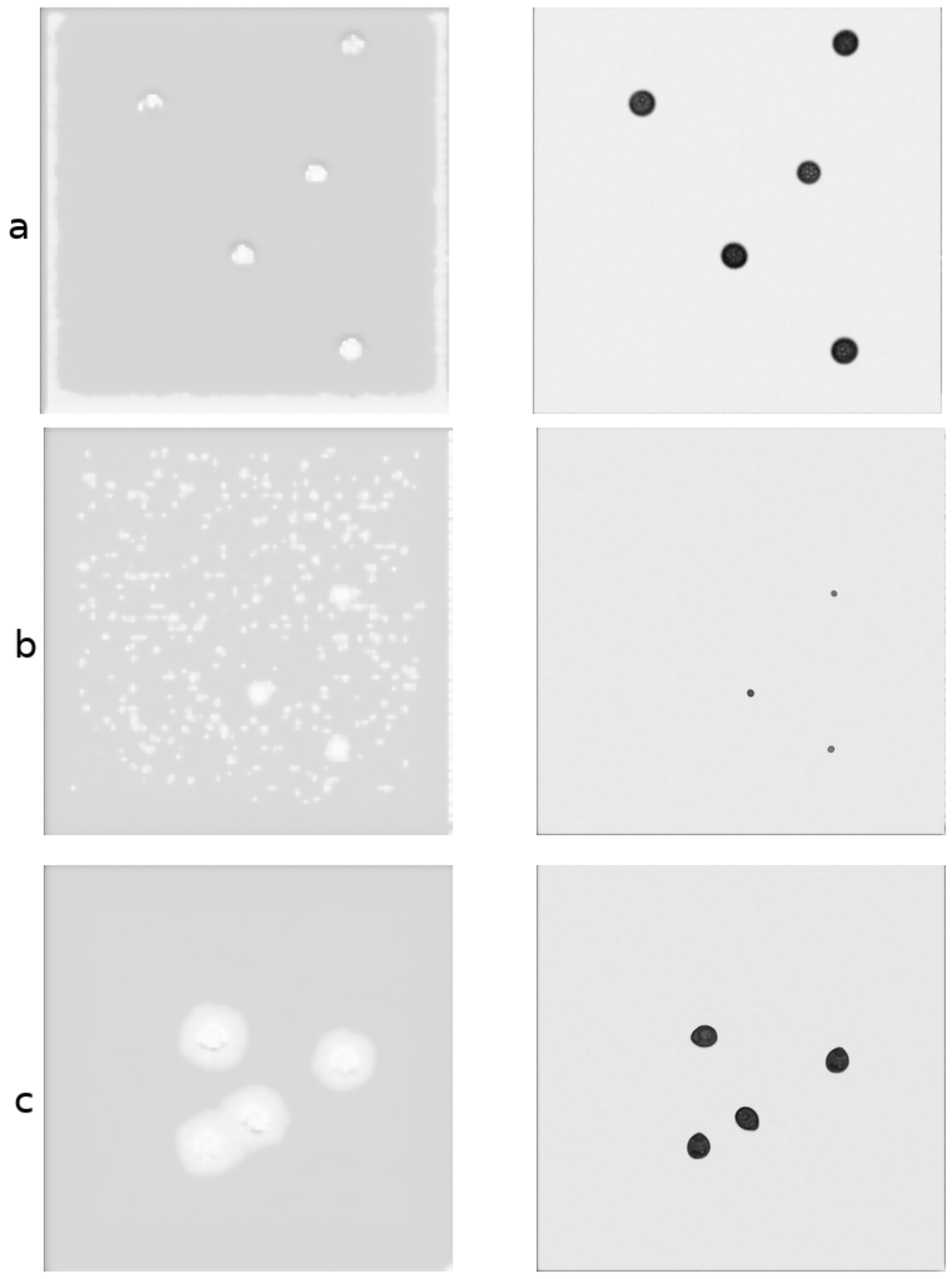

Figure 47: Images from a pollen simple sub-dataset after processing with Extremal Perturbation. Left image - image after EP; Right image - original image

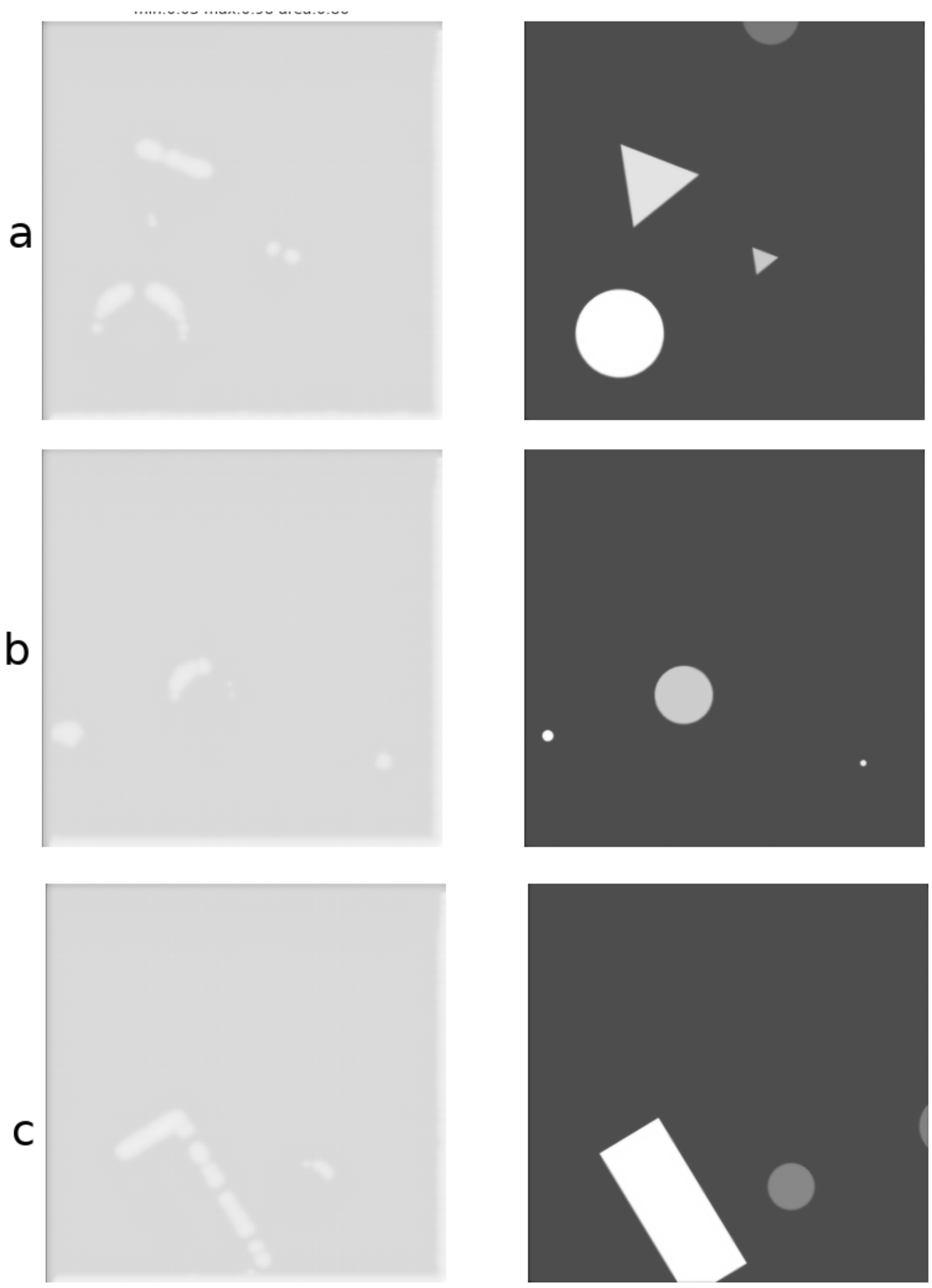

Figure 48: Images from a geometrical mixed sub-dataset after processing with Extremal Perturbation. Left image - image after EP; Right image - original image

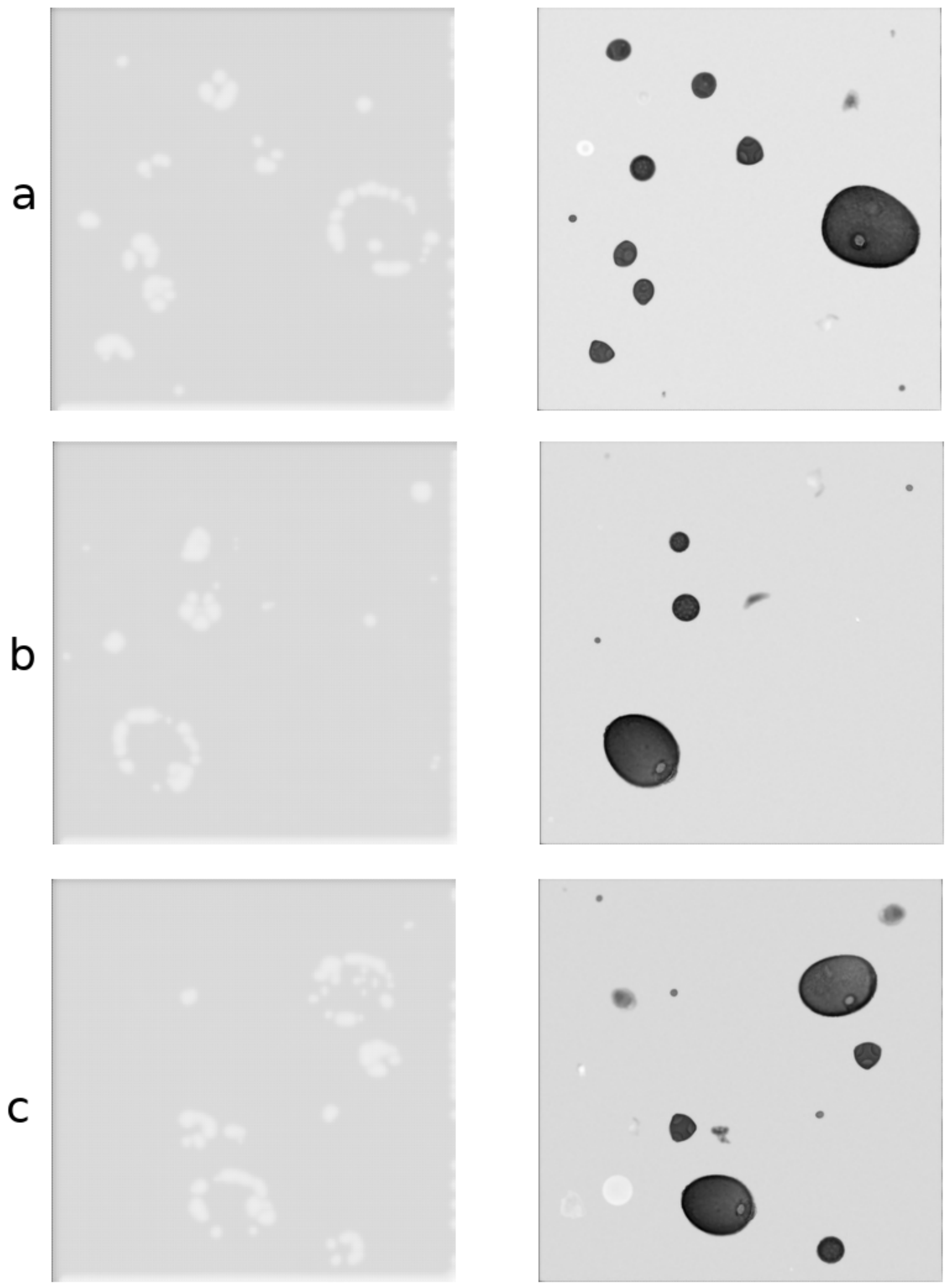

Figure 49: Images from a pollen mixed sub-dataset after processing with Extremal Perturbation. Left image - image after EP; Right image - original image

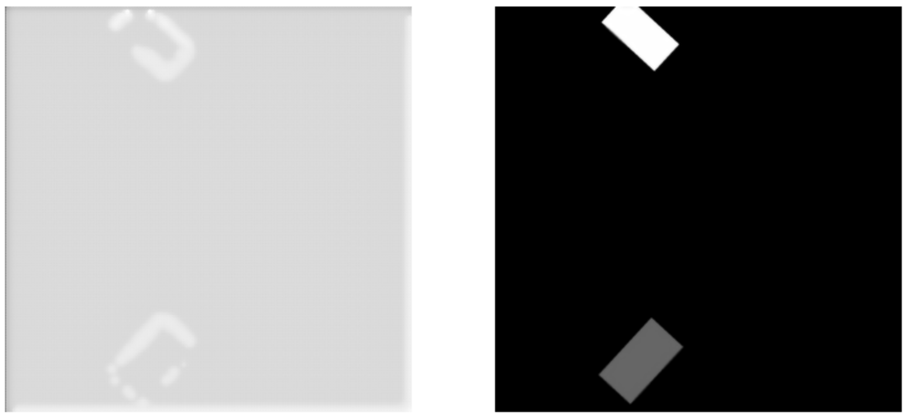
Figure 50: The image from the geometrical simple dataset after the Extremal perturbation method processing

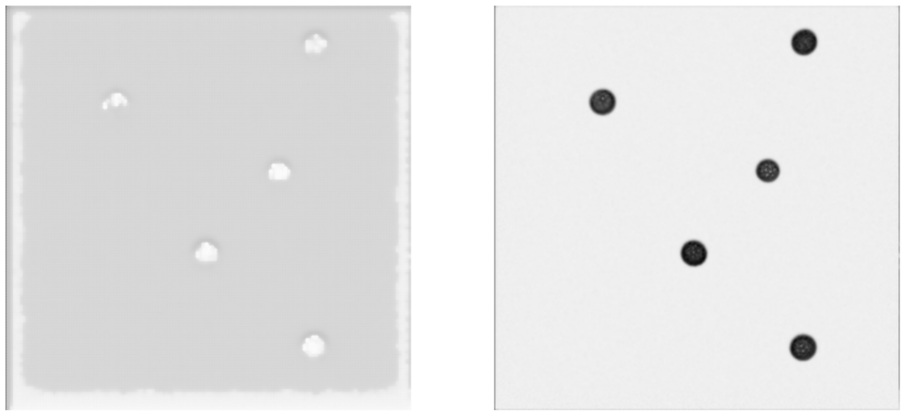
Figure 51: The image from the pollen simple dataset after the Extremal perturbation method processing

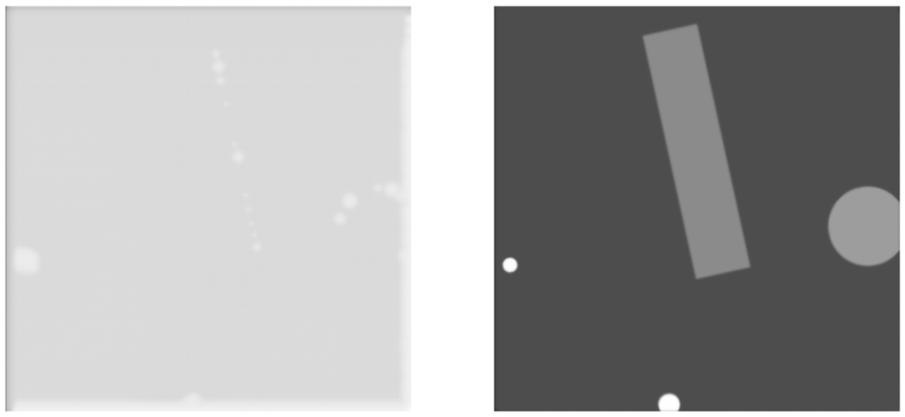
Figure 52: The image from the geometrical mixed dataset after the Extremal perturbation method processing

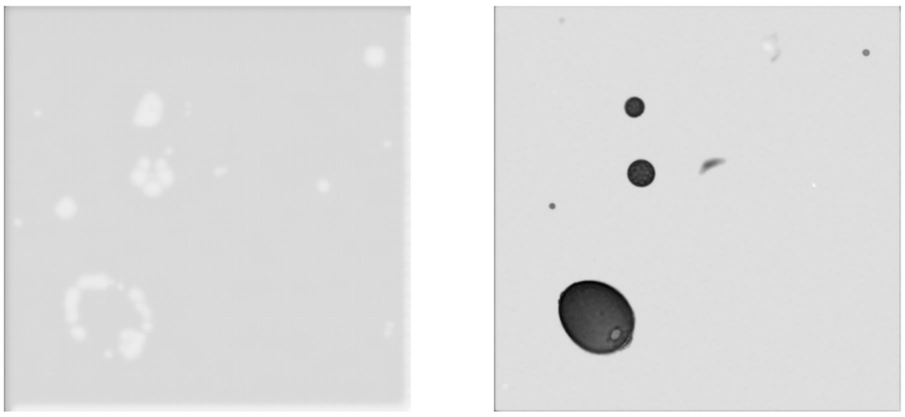

Figure 53: The image from the pollen mixed dataset after the Extremal perturbation method processing

D Results of Localisation

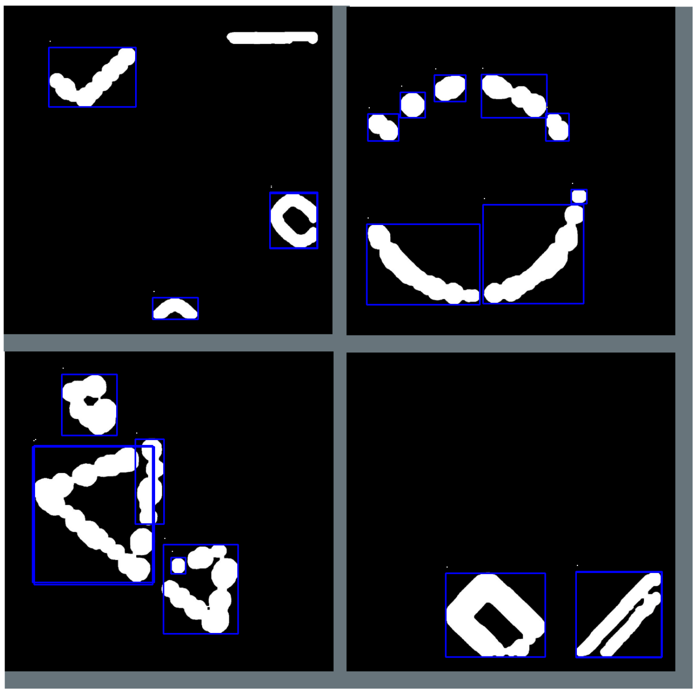

Figure 54: The sample of 4 images from geometric simple sub-dataset after applying the approach to self-supervised object localisation through deep learning based classification

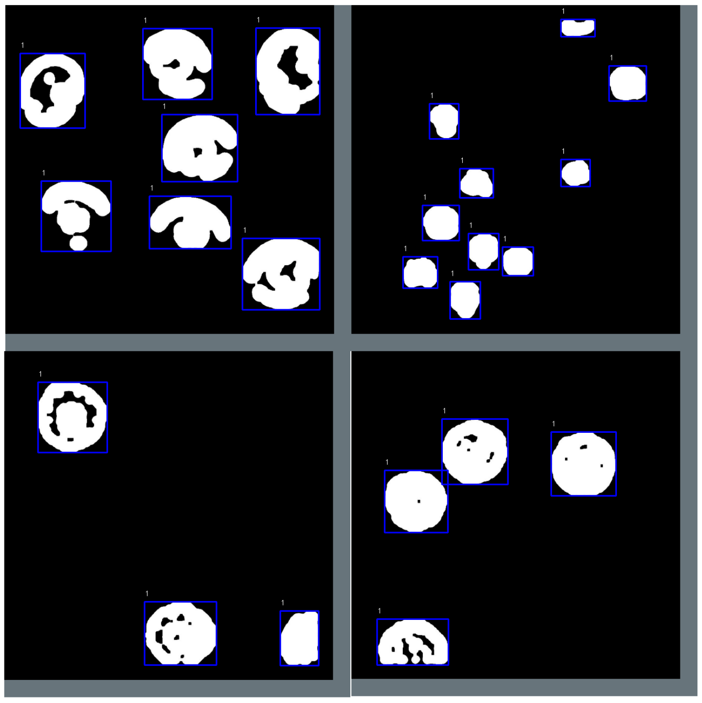

Figure 55: The sample of 4 images from pollen simple sub-dataset after applying the approach to self-supervised object localisation through deep learning based classification

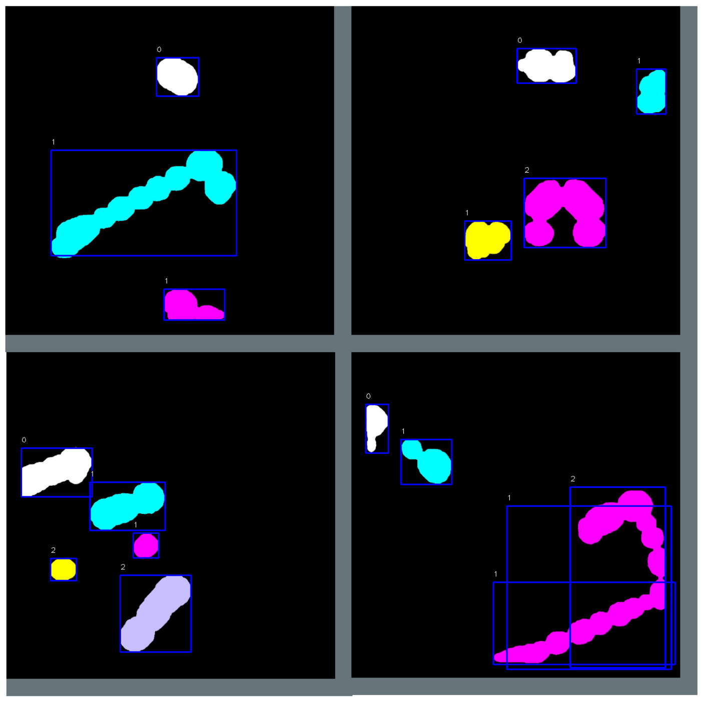

Figure 56: The sample of 4 images from geometric mixed sub-dataset after applying the approach to self-supervised object localisation through deep learning based classification

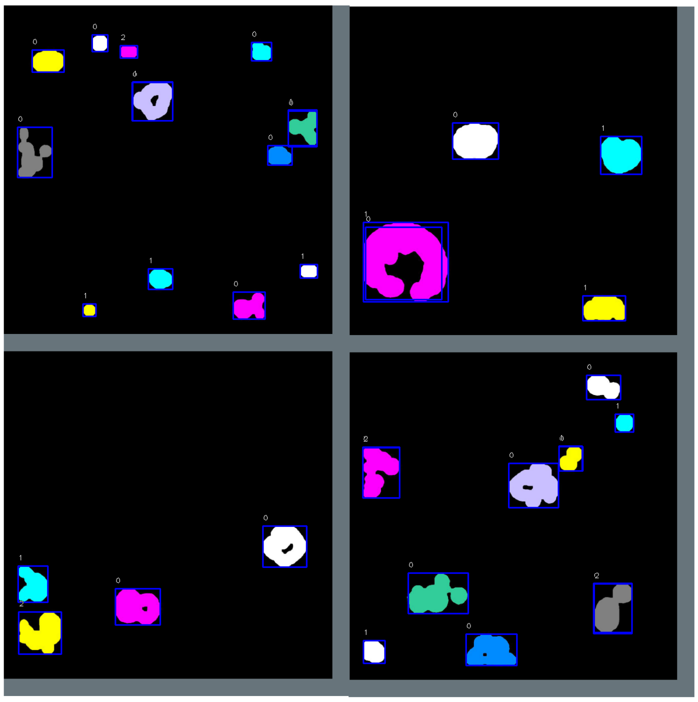
Figure 57: The sample of 4 images from pollen mixed sub-dataset after applying the approach to self-supervised object localisation through deep learning based classification

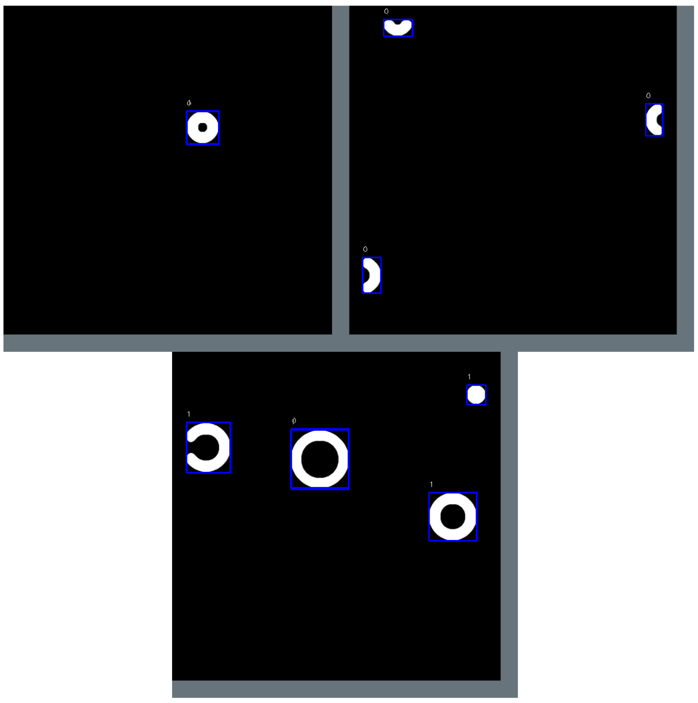
Figure 58: The sample of images from GT geometric simple sub-dataset after applying the model trained on EP dataset

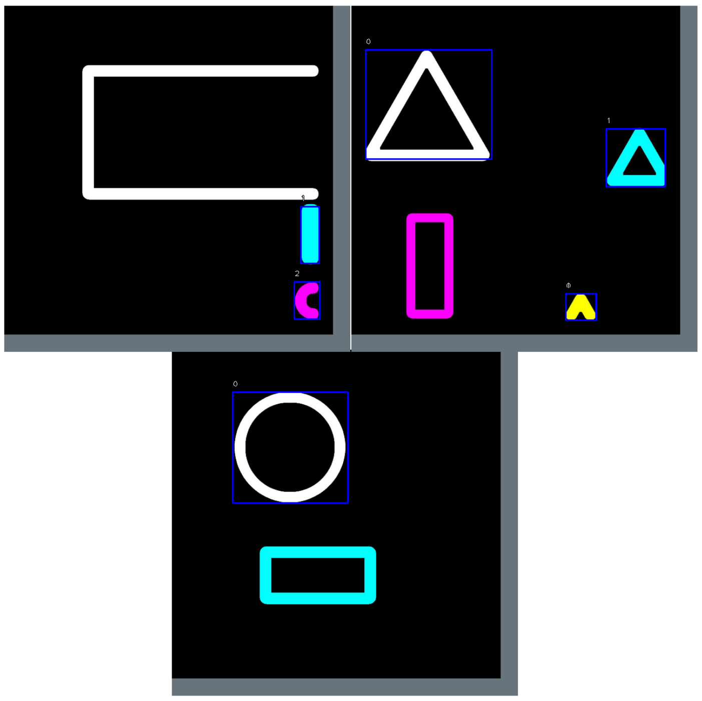

Figure 59: The sample of images from GT geometric mixed sub-dataset after applying the model trained on EP dataset

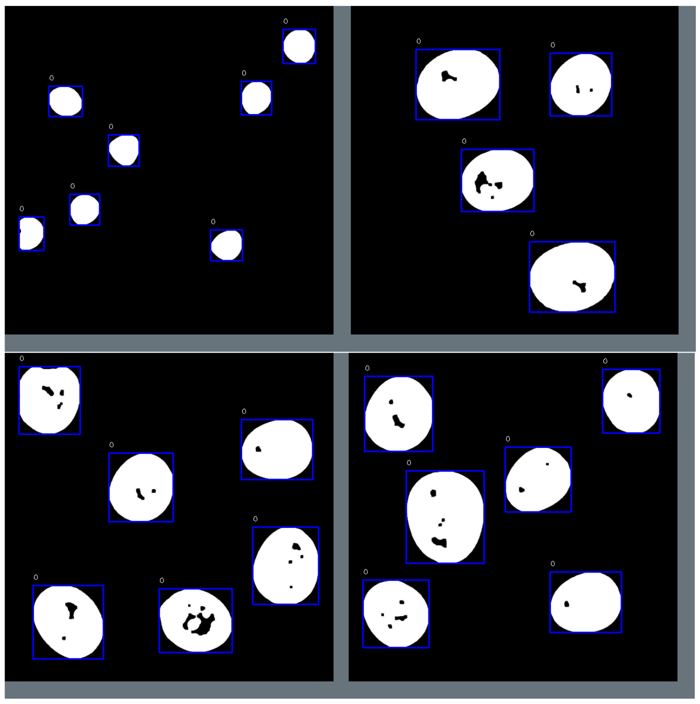

Figure 60: The sample of images from GT pollen simple sub-dataset after applying the model trained on EP dataset

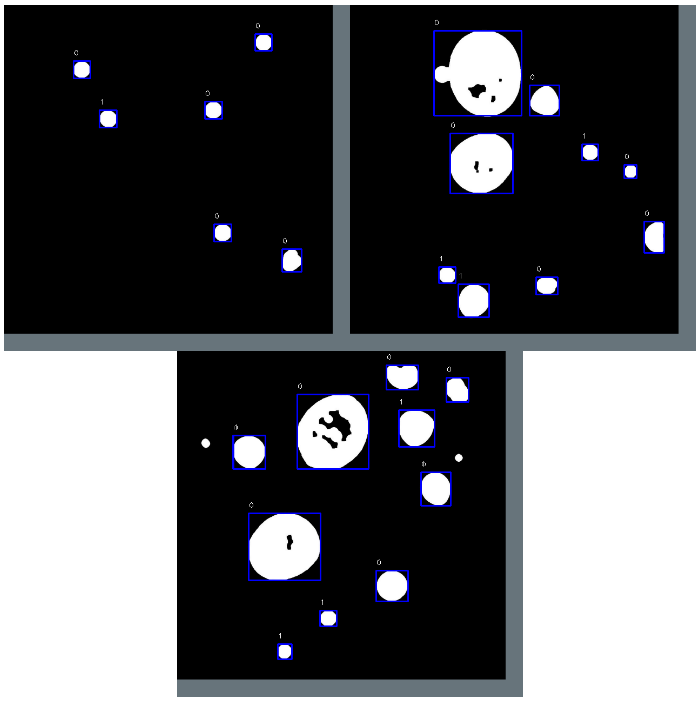
Figure 61: The sample of images from GT pollen mixed sub-dataset after applying the model trained on EP dataset

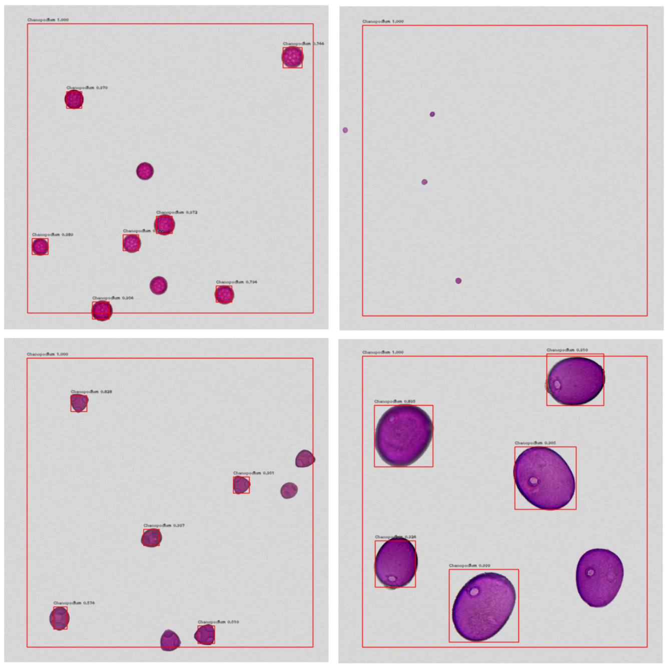
Figure 62: The sample of images from GT datasets with original annotations pollen simple sub-dataset after applying the model trained on EP dataset

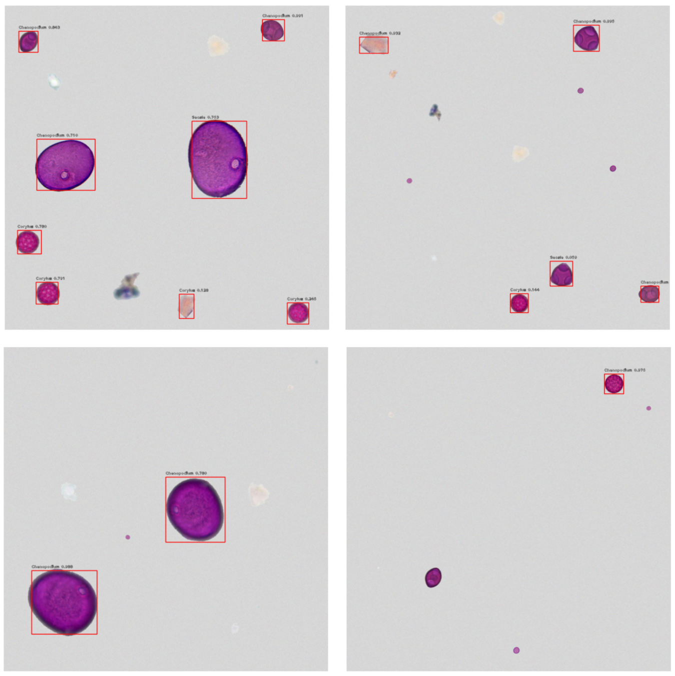

Figure 63: The sample of images from GT datasets with original annotations pollen mixed sub-dataset after applying the model trained on EP dataset

References

[1] waspinator/pycococreator: Helper functions to create coco datasets. `https://github.com/waspinator/pycococreator/`, 2018. (Accessed on 02/13/2021).

[2] 3.3. metrics and scoring: quantifying the quality of predictions - scikit-learn 0.24.1 documentation. `https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics`, 2020. (Accessed on 02/14/2021).

[3] sklearn.metrics.f1_score - scikit-learn 0.24.1 documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`, 2020. (Accessed on 02/08/2021).

[4] Welcome to torchray — torchray beta documentation. `https://facebookresearch.github.io/TorchRay/`, 2020. (Accessed on 02/12/2021).

[5] Introduction · captum. `https://captum.ai/docs/introduction`, 2021. (Accessed on 02/11/2021).

[6] Nvlink & nvswitch:advanced multi-gpu systems — nvidia. `https://www.nvidia.com/en-us/data-center/nvlink/`, 2021. (Accessed on 02/03/2021).

[7] yhenon/pytorch-retinanet: Pytorch implementation of retinanet object detection. `https://github.com/yhenon/pytorch-retinanet`, 2021. (Accessed on 02/14/2021).

[8] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. *CoRR*, abs/1810.03292, 2018.

[9] B. Aggarwal, A. Acharya, and A. Laghari. A survey on image datasets for computer vision and deep learning convolutional neural network tasks. 10 2020.

[10] I. Ahmed. Pneumonia detection: Fine tuning and cam. `https://www.kaggle.com/ibtesama/pneumonia-detection-fine-tuning-and-cam`, 2019. (Accessed on 02/20/2021).

[11] I. Aizenberg, N. N. Aizenberg, and J. P. Vandewalle. *Multi-valued and universal binary neurons: Theory, learning and applications*. Springer Science & Business Media, 2013.

[12] A. Anandkumar, R. Ge, D. J. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *CoRR*, abs/1210.7559, 2012.

[13] B. Andres. Machine learning 1 lectures. In *Machine Learning for Computer Vision. TU Dresden*, 2020. (Accessed on 01/25/2021).

[14] R. Arandjelovic and A. Zisserman. Look, listen and learn. *CoRR*, abs/1705.08168, 2017.

[15] N. Arun, N. Gaw, P. Singh, K. Chang, M. Aggarwal, B. Chen, K. Hoebel, S. Gupta, J. Patel, M. Gidwani, J. Adebayo, M. D. Li, and J. Kalpathy-Cramer. Assessing the (un)trustworthiness of saliency maps for localizing abnormalities in medical imaging. *medRxiv*, 2020.

[16] T. Baltrušaitis, C. Ahuja, and L.-P. Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[17] J. Brownlee. Difference between classification and regression in machine learning. *Machine Learning Mastery*, 25, 2017.

[18] M. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *CoRR*, abs/1612.03365, 2016.

[19] L. Chen, P. Bentley, K. Mori, K. Misawa, M. Fujiwara, and D. Rueckert. Self-supervised learning for medical image analysis using image context restoration. *Medical Image Analysis*, 58:101539, 2019.

[20] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[21] G. E. Dahl, N. Jaitly, and R. Salakhutdinov. Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[22] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[23] J. Dean. The deep learning revolution and its implications for computer architecture and chip design. *CoRR*, abs/1911.05289, 2019.

[24] R. Dechter. Learning while searching in constraint-satisfaction problems. 1986.

[25] R. Dechter. Learning while searching in constraint-satisfaction-problems. pages 178–185, 01 1986.

[26] A. J. DeGrave, J. D. Janizek, and S.-I. Lee. Ai for radiographic covid-19 detection selects shortcuts over signal. *medRxiv*, 2020.

[27] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[28] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. 2009.

[29] A.-J. Dorne. Variation in seed germination inhibition of chenopodium bonus-henricus in relation to altitude of plant growth. *Canadian journal of Botany*, 59(10):1893–1901, 1981.

[30] V. Erdogan and S. A. Mehlenbacher. Interspecific hybridization in hazelnut (corylus). *Journal of the American Society for Horticultural Science*, 125(4):489–497, 2000.

[31] A. K. S. et. all. Nanodiamonds from laser-induced shock compression of polystyrene: Extraction under way - publications repository - helmholtz-zentrum dresden-rossendorf, hzdr. `https://www.hzdr.de/db/!Publications?pNid=head&pSelMenu=0&pSelTitle=30247`. (Accessed on 02/01/2021).

[32] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, volume 2, page 8, 2014.

[33] R. Fong, M. Patrick, and A. Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2950–2958, 2019.

[34] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

[35] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[36] K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982.

[37] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[38] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[39] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[40] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[41] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[42] P. Goyal, D. Mahajan, A. Gupta, and I. Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6391–6400, 2019.

[43] I. A. Grigorevich and L. V. Grigorevich. Cybernetics and forecasting techniques. 1967.

[44] B. Guthier. Machine learning 2 lectures. winter semester 2018/19, tu dresden. (Accessed on 01/25/2021), 2018.

[45] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[46] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[47] S. Hu, X. Chen, and X. Tong. Point sets joint registration and co-segmentation. *The Visual Computer*, 35(12):1841–1853, 2019.

[48] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[49] A. G. Ivakhnenko. Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*, (4):364–378, 1971.

[50] S. Jawed, J. Grabocka, and L. Schmidt-Thieme. Self-supervised learning for semi-supervised time series classification. In H. W. Lauw, R. C.-W. Wong, A. Ntoulas, E.-P. Lim, S.-K. Ng, and S. J. Pan, editors, *Advances in Knowledge Discovery and Data Mining*, pages 499–511, Cham, 2020. Springer International Publishing.

[51] D. Jayaraman and K. Grauman. Learning image representations equivariant to ego-motion. *CoRR*, abs/1505.02206, 2015.

[52] H. Jiang, G. Larsson, M. M. G. Shakhnarovich, and E. Learned-Miller. Self-supervised relative depth learning for urban scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–35, 2018.

[53] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[54] L. Jing, X. Yang, J. Liu, and Y. Tian. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv preprint arXiv:1811.11387*, 2018.

[55] J. Johnson and A. Karpathy. Convolutional neural networks (cnns / convnets). `http://cs231n.github.io/convolutional-networks/` last visited 2019-09-30.

[56] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *CoRR*, cs.AI/9605103, 1996.

[57] D. Kim, D. Cho, D. Yoo, and I. S. Kweon. Learning image representations by completing damaged jigsaw puzzles. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 793–802. IEEE, 2018.

[58] Y. Kim, S. Wiseman, and A. M. Rush. A tutorial on deep latent variable models of natural language. *CoRR*, abs/1812.06834, 2018.

[59] A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1920–1929, 2019.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[62] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6874–6883, 2017.

[63] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[64] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[65] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 667–676, 2017.

[66] D. Li, W.-C. Hung, J.-B. Huang, S. Wang, N. Ahuja, and M.-H. Yang. Unsupervised visual representation learning by graph-based consistent constraints. In *European Conference on Computer Vision*, pages 678–694. Springer, 2016.

[67] D. Y. Li Deng. *Deep Learning: Methods and Applications*, volume 7. 2014.

[68] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[69] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[70] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[71] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[72] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[73] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[74] V. A. Mahendran A. Salient deconvolutional networks. *Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9910. Springer, Cham*, 2016.

[75] N. Manzini. Single hidden layer neural network. `https://www.nicolamanzini.com/single-hidden-layer-neural-network/`, 11 2019. (Accessed on 01/31/2021).

[76] U. Markwardt. Hpc-da - compendium - foswiki. `https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/HPCDA`, 05 2020. (Accessed on 02/03/2021).

[77] U. Markwardt. Power9 - compendium - foswiki. `https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/Power9`, 02 2020. (Accessed on 02/03/2021).

[78] U. Markwardt. Systemtaurus - compendium - foswiki. `https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/SystemTaurus`, 01 2021. (Accessed on 02/03/2021).

[79] N. Miailhe and C. Hodes. The third age of artificial intelligence. *Field actions science reports. The Journal of Field Actions*, (Special Issue 17):6–11, 2017.

[80] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[81] T. Mitchell. Machine learning textbook. `http://www.cs.cmu.edu/~tom/mlbook.html`, 1997. (Accessed on 01/25/2021).

[82] M. H. Modarres, R. Aversa, S. Cozzini, R. Ciancio, A. Leto, and G. P. Brandino. Neural network for nanoscience scanning electron microscope image recognition. *Scientific reports*, 7(1):1–12, 2017.

[83] K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[84] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[85] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9359–9367, 2018.

[86] R. Padilla, S. L. Netto, and E. A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[87] N. Passalis and A. Tefas. Probabilistic knowledge transfer for deep representation learning. *CoRR*, abs/1803.10837, 2018.

[88] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017.

[89] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[90] J. M. Pena, J. A. Lozano, P. Larranaga, and I. Inza. Dimensionality reduction in unsupervised learning of conditional gaussian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):590–603, 2001.

[91] V. Petsiuk, A. Das, and K. Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[92] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. *arXiv preprint arXiv:1506.06204*, 2015.

[93] M. Raghu and E. Schmidt. A survey of deep learning for scientific discovery. *arXiv preprint arXiv:2003.11755*, 2020.

[94] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[95] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[96] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[97] Z. Ren and Y. J. Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–771, 2018.

[98] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[99] M. Robnik-Šikonja and M. Bohanec. *Perturbation-Based Explanations of Prediction Models*, pages 159–175. Springer International Publishing, Cham, 2018.

[100] S. Saha. A comprehensive guide to convolutional neural networks — the eli5 way — by sumit saha — towards data science. `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b11` 12 2018. (Accessed on 02/01/2021).

[101] N. Sayed, B. Brattoli, and B. Ommer. Cross and learn: Cross-modal self-supervision. In *German Conference on Pattern Recognition*, pages 228–243. Springer, 2018.

[102] R. Schlegel. Rye (secale cereale l.)—a younger crop plant with bright future. *Genetic resources, chromosome engineering, and crop improvement*, 2:365–394, 2006.

[103] J. Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[104] H. Schulz. High performance computing at hzdr - helmholtz-zentrum dresden-rossendorf, hzdr. `https://www.hzdr.de/db/Cms?pOid=12231&pNid=852`, 04 2020. (Accessed on 02/03/2021).

[105] T. J. Sejnowski. *The Deep Learning Revolution.* Mit Press, 2nd edition, 2018.

[106] T. J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020.

[107] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[108] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2nd international conference on learning representations, iclr 2014. Jan. 2014. 2nd International Conference on Learning Representations, ICLR 2014 ; Conference date: 14-04-2014 Through 16-04-2014.

[109] R. Setiono and W. K. Leow. Fernn: An algorithm for fast extraction of rules fromneural networks. *Applied Intelligence*, 12(1–2):15–25, Jan. 2000.

[110] S. Seurig, P. Steinbach, N. Scherf, and I. Röder. Simulated pollen microscope slides for segmentation/bounding box regression and classification, Oct. 2020.

[111] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.

[112] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[113] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017.

[114] F. T. M. Spieksma, J. Corden, M. Detandt, W. Millington, H. Nikkels, N. Nolard, C. Schoenmakers, R. Wachter, L. De Weger, R. Willems, et al. Quantitative trends in annual totals of five common airborne pollen types (betula, quercus, poaceae, urtica, and artemisia), at five pollen-monitoring stations in western europe. *Aerobiologia*, 19(3):171–184, 2003.

[115] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[116] P. Steinbach. Nomenclature.png (534×527). `https://gitlab.hzdr.de/haicu/theses/thesis_self-supervised_localisation/-/raw/master/rsc/notes/Nomenclature.png`. (Accessed on 02/19/2021).

[117] O. Stretcu and M. Leordeanu. Multiple frames matching for object discovery in video. In *BMVC*, volume 1, page 3, 2015.

[118] M. Usama, J. Qadir, A. Raza, H. Arif, K. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha. Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE Access*, 7:65579–65615, 2019.

[119] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[120] R. Vargas, A. Mosavi, and R. Ruiz. Deep learning: A review. *Advances in Intelligent Systems and Computing*, 5, 06 2017.

[121] B. Wallace and B. Hariharan. Extending and analyzing self-supervised learning across domains. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 717–734, Cham, 2020. Springer International Publishing.

[122] A. Wan, L. Dunlap, D. Ho, J. Yin, S. Lee, H. Jin, S. Petryk, S. A. Bargal, and J. E. Gonzalez. Nbdt: Neural-backed decision trees, 2021.

[123] H. Wang, B. Raj, and E. P. Xing. On the origin of deep learning. *CoRR*, abs/1702.07800, 2017.

[124] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[125] K. Yang, Y. Cao, Y. Zhang, M. Tang, D. Aberg, B. Sadigh, and F. Zhou. Self-supervised learning and prediction of microstructure evolution with recurrent neural networks. *arXiv preprint arXiv:2008.07658*, 2020.

[126] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

[127] L. Zhang, L. Lin, X. Liang, and K. He. Is faster r-cnn doing well for pedestrian detection? In *European conference on computer vision*, pages 443–457. Springer, 2016.

[128] Q. Zhang, L. T. Yang, Z. Chen, and P. Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018.

[129] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[130] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.

[131] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.