Universiteit Leiden
The Netherlands

Informatica & Economie

Organizing and classifying historical butterfly collections from Indonesia

Casper Heijnen & Koen van Neerbos

Supervisors:
Prof. Dr. Ir. Fons J Verbeek - LIACS
Dr. Rutger Vos - Naturalis

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/02/2018

Abstract

The number of butterfly taxonomists in the world is decreasing. In order to preserve this knowledge a classification program for butterflies was build. During the 1930s a large dataset of butterflies was collected from Indonesia and stored in Naturalis. This dataset was used to test the classification program. All the images were digitized by Naturalis and saved on Flickr. A MonetDataBase was build to store these images in, MonetDataBase is a scalable system, hence it was used. In this research Histogram of Oriented Gradient were used to decrease the size of the butterfly images. Three different classifiers where tested: Naïve bayes, K-Nearest-Neigbour and Support Vector Machine. SVM showed the best results. 95% Of the butterflies can be classified correctly if the training contains more than 40 examples per species. The pipeline that was created for this thesis can be used for all kind of classification problems.

Acknowledgements

This bachelor thesis would not have been possible without several people. We would like to thank our supervisor Prof.dr.ir. F.J. Verbeek for his time and support. We also like to thank Rutger Vos for providing us the dataset . A big thanks goes out to Leon Helwerda who helped us a lot with the python coding. We would like to thank Yuanhao Guo who helped a lot at the beginning of our research with harvesting of the dataset. Finally we would like to thank our families and friends for supporting and motivate us.

Contents

Chapter 1

Introduction

Taxonomy is the science of defining and naming groups of biological organisms on the basis of shared characteristics. [1]

Taxonomist is one of the older professions in the world. In Egypt, wall paintings where found that clearly shows societies communicating the uses of different species (1500BC). They needed taxonomy to let other people know which plants were poisonous and which animals were not dangerous. One of the most important taxonomist in the history is Charles Darwin who changed the scientific way of taxonomy with his book: "Charles Darwin, the origin of species" (1859).

The catalogue of life tries to document all the species in the world. The catalogue describes the class insects. The Lepidoptera is the fourth largest insect order in the world according to the catalogue of life. The Lepidoptera order contains butterflies and moths [2]. Every year new species of the Lepidoptera order are found.

During the 1930s a group of biologists went on an expedition in Indonesia. During this expedition they collected a lot of plants and animals including butterflies. The butterflies they collected were sent too the Netherlands and are currently stored in the Naturalis biodiversity center. Naturalis is a research center on biodiversity as well as a national museum of natural history. Their complete collection comprises roughly 36 million items. [3]

Previously the butterflies were stored in giant drawer cabinets. As you can see on figure 1.1. For a long time, this collection was collecting dust. The collection of butterflies is partly named by species by taxonomists.

The process of converting information into a digital format is called digitization. This information could be an object, sound, image, document or signal. Since 2011 Naturalis is working on digitizing the collection. With 36 million items in their collection this will be a time consuming job. At the start of this digitizing project it was a very time consuming job to digitize just one item. Over the past years the digitization process has improved a lot. The online collection of butterflies is growing rapidly [4]. Naturalis does the digitizing to make the collections available for everyone with a computer and internet connection.

Figure 1.1: Drawers with butterflies [5]

Classification is a general process related to categorization, the process in which ideas and objects are recognized, differentiated, and understood. [7] Classification is the problem of identifying a number of attributes of a new observation. With a training set of attributes whose category is known a classifier can be build that identifies the new set of attributes correctly and can put it in the right category. During our thesis classification will be applied in order to identify the species of the butterflies.

The number of taxonomists around the world is declining. If this decline continues a lot of knowledge would disappear. A way to harvest the knowledge is to build a classifier.

A previous project started by Mengke Li in order to investigate the efficiency of the classifier on butterflies. During her Research the data set was small. Too small to make the classifier work proper. [9] During the last couple of years, the digitization process has improved a lot and there are 3 times as many butterfly pictures online. Mengke could classify on genus level. The genus level is one level above specie level in the hierarchy of biological classification's eight major taxonomic ranks. In this research an attempt will be done to classify on specie level.

Figure 1.2 shows the classification process. There are two aspects in this research: Software engineering and Datamining. The software engineering part contains python, database and writing scripts. Figure 1.2 boxes one till eight asked for software engineering. Box nine asked for datamining.

A quick overview of the pipeline in figure 1.2 shows 9 steps to perform. The butterflies arrived at Naturalis, they have digitalized them by taking pictures (1). These pictures and metadata are stored in Flickr (2). When the pictures are available the image harvesting can start (3). Additional information is stored in a seperate file (4). Together with this extra information about the butterflies these can be stored in MonetDB (5/6). To start the classification process the data should be cleaned (7). Next feature extraction will be executed (8). With the clean data and features, machine learning can be applied (9). Of course step 5 should be done only once. After all, if the database is created all new data can be simply added.

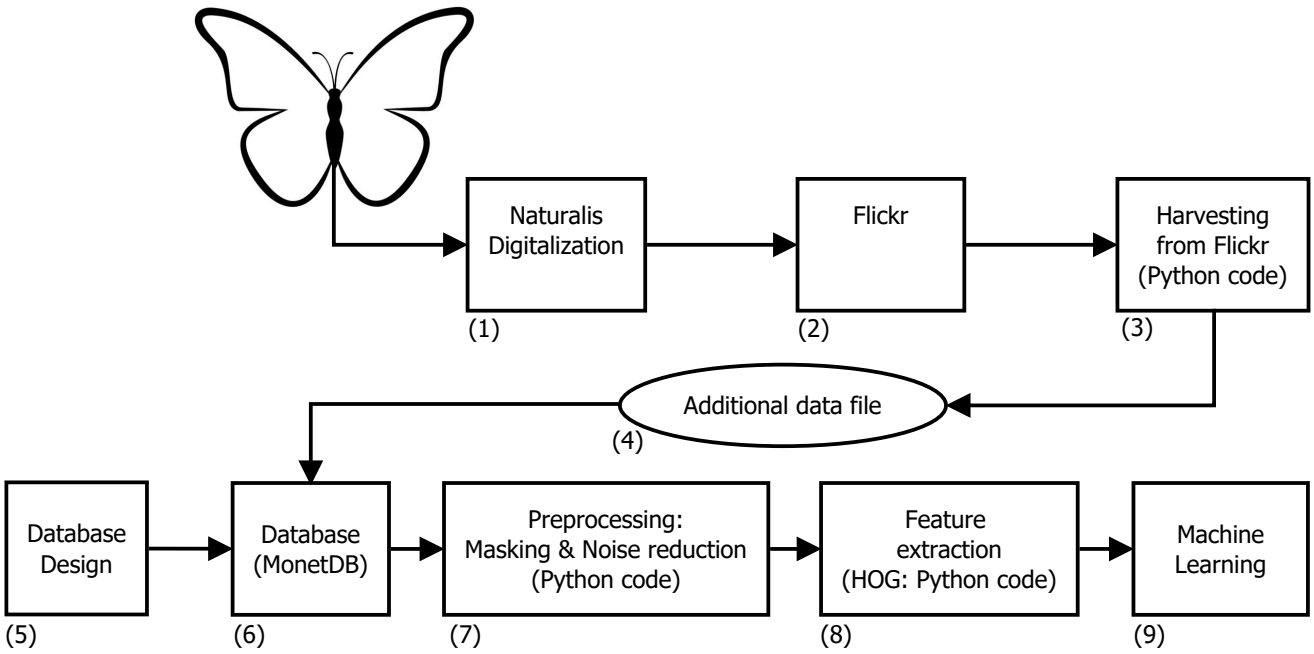

Figure 1.2: The pipeline of the classification process.

1.1 Division of labor

This thesis project is performed by two students. During this thesis the division of labor will be as following. Figure 1.2 boxes three, four, five and six will be done by KvN, boxes seven eight and nine will be done by CH. There will be intensive collaboration during this thesis work. This collaboration will mainly be in the form of the machine learning experiments (box nine).

1.2 Questions

During this project the research question will be:

- Is it possible to classify the butterflies on the level of specie ?

This project should answer the following subquestions:

- Is it possible to classify the Lepidoptera on specie level?
- Which classifier is the most successful for this problem?
- Are there any boundary conditions to this classifier?(number of examples per specie?)
- Can the classification process be automated?
- Is MonetDB qualified for this problem?

1.3 Thesis Overview

This chapter contains the introduction in which the subject and goals will be explained, Chapter 2 includes the Design. This shows what the workflows how we want to solve the problem.

Chapter 3 describes the materials and methods used for the problem

Chapter 4 contains the implementation of the methods discussed in Chapter 3. Chapter 5 discusses the experiments, it shows what kind of experiments are done and why these experiments are done.

Chapter 6 explains the results that are found in the experiments. Furthermore it describes what kind of things can also be tested in the future as well as how this research can be extended. At last it is discussed how this research can be used in other context.

Chapter 2

Design

The images of the butterflies are published on the online photo service Flickr. On Flickr the pictures are annotated with standardized tags with for example the species name [10]. All the pictures have got the same set up. A butterfly is folded and laid on it side on paper that includes a ruler. In the right-upper corner there is some additional information about the butterfly that is shown. This is shown in figure 2.1. Today, the Naturalis-account on Flickr contains about 1842 pictures of butterflies. The photos are shot at Naturalis with a Nikon D600. The images are 4,8 MB and have a size of 6016 px by 4016 px.

Figure 2.1: An example how a butterfly is stored in Flickr

2.1 Database

This section will mainly about figure 1.2 box six.

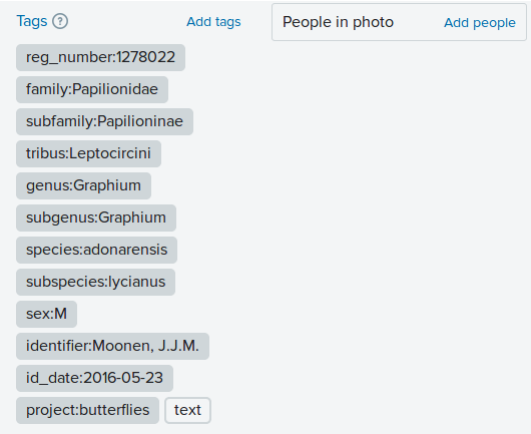

Figure 2.2: Photo tags inside Flickr

The images on Flickr are saved with tags. An example can be found in figure 2.2. The tags contain information about the butterfly. In order to make a complete database this information has to be stored. Naturalis handed us another database which contained information about the location of discovery. All the information of the tags and the database Naturalis has got can be merged into one database. If this database was created with all these information a database with 12 columns would be created, this is not conducive to readability. In order for the database to be better readable a design was made were the 4 major groups of information are split. Taxonomy, place of discovery, photo and collecting. The database place and collecting were not very interesting during this thesis so these were separated.

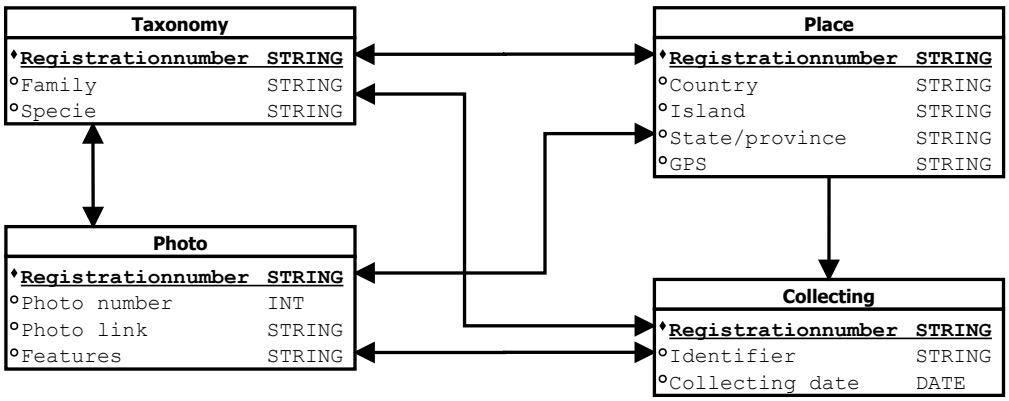

Figure 2.3: Database model

Figure 2.3 shows the model of the database. The most important tag to save is the title of the photo and the re_number. The database from Naturalis contains information about the butterfly that is not saved on Flickr. The Flickr tags and Naturalis database can be linked. Both the Flickr tags and Naturalis database contain the registration number. Once linked a new database can be build that contains all the information from Flickr and the Naturalis database. This database needs to be scalable and fast. The images of the butterflies are very large. Therefor a link to the picture will be saved. When the image features are calculated they can be stored in the database. When all this information is stored in the database the machine learning can start. Of course later on the database can be expanded with additional information about the butterflies and new butterlfies.

2.2 Workflow feature extraction

This section will include some elaboration on figure 1.2. Box three, seven and eight will be appointed below.

The images stored on Flickr are on average 4MB per photo. With more than 1800 images the dataset is roughly 21 GB in size. This dataset could be used but it would take a lot of time to experiment. The decision was made to reduce the amount of data in the dataset without losing crucial information. Figure 2.1 shows that there is a lot of redundant data. The only thing that is needed is the butterfly itself. So the information card in the upper right corner and even the line paper in the background is redundant. All these additional information is not needed for our research.

To reduce the amount of data the redundant parts of the images are being deleted so only the butterfly remains.

Feature extraction is reducing the amount of data needed to describe a large set of data. There are a lot of feature extraction methods.

The butterfly is our region of interest (ROI). There are a lot of feature extraction techniques and one of them will be used on the butterfly. A few examples of features are SIFT, SURF, GLOH and HOG.

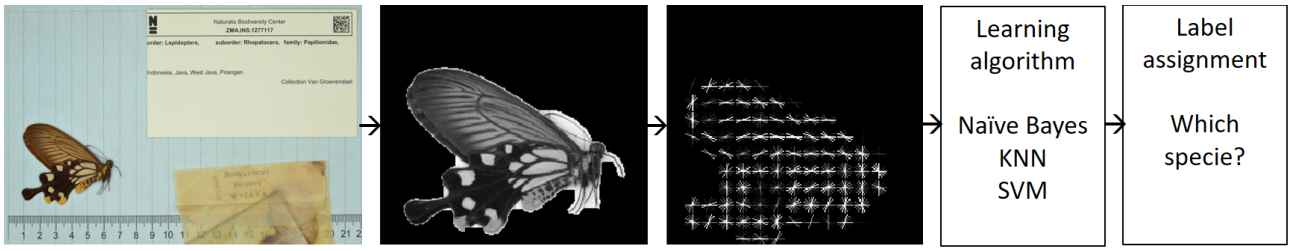

Figure 2.4: The pipeline how to classify the butterflies.

Figure 2.4 shows the pipeline of how to do the classification. This pipeline contains the following steps. At first all the redundant data must be eliminated. To do this, the butterfly has to be cut out of the whole picture. This will be the input-image. The next step in preprocessing will be resizing the images to one size. So all images will have the same size. After the preprossessing is done, the features can be extracted. Feature extraction is reducing the amount of resources needed to describe a large set of data. After the feature extraction the classifying algorithms can be executed. A few examples of classifiers are: k-nearest neighbour, J48, Decision tree, Support Vector Machine, Naïve bayes and Kstar

2.3 Learning algorithm

With the features of the images saved in the database a learning algorithm can be applied. At first the algorithm will be tested on a labeled dataset. In a labeled dataset the features of an image are linked to the family and species of the butterfly. With the labeled dataset it is possible to find the best algorithm for this problem. Once the best algorithm is found the label assignment can start on an unlabeled dataset. For example when Naturalis drops a new image in the database without any tags. The algorithm will provide the image with a specie name.

2.4 Workflow Labels

A supervised learning algorithm requires a labeled dataset. In order to fill the database it is important to link the file name to the specie that is on the sheet. The specie on the image is saved in the tags on Flickr. These tags need to be connected to the filename of the image. The species name of the butterfly can be seen on the picture. If it is not possible to connect the tags to the file name it would be a possibility to read the species name from the image with a image-to-text program. In the top right of the image a datamatrix can be found. Figure 2.5 show such a datamatrix. A datamatrix is a square or rectangle with cells. These cells can be either one or zero (black or white). In a datamatrix data can be saved. The data can be text or numerical. It is a two dimensional barcode. In case of the image of the butterfly the registration number is saved in the datamatrix. For example $ZMA.INS.XXXXXXXX$ (where x is a number).

Figure 2.5: A datamatrix that can be found in the upper-right of a picture

Chapter 3

Materials and methods

In this chapter the materials and methods will be discussed. In chapter 4 the implementation and use of these materials and methods will be discussed.

3.1 Python

Pyhton is a programming language. Python is widely used high level programming language. Python is designed by Guido van Rossum. Guido wanted a programming language that had a better readability and fewer lines of code than other programming languages such as Java and C++. [11] Python is platform independent. There are some modules and function that are platform based so with these modules and functions it would become platform dependent. In this research the python code is platform independent. Pyhton was used in this thesis because MonetDB has a programming interface for python.

3.2 Harvest

On the internet a Flickr image harvester is posted [1]. This harvester is written in python. Once the images were downloaded the program also made a .db file. In this file the tags where saved. The .db file did not contain the file name of the image.

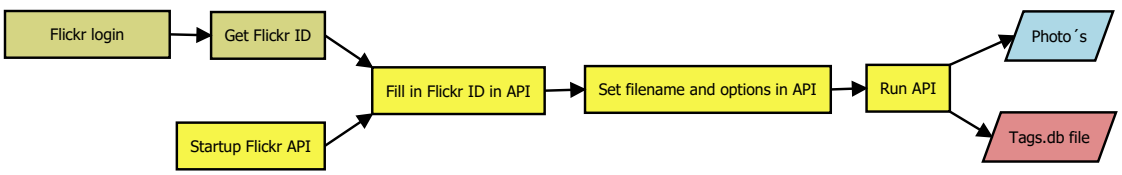

Figure 3.1: From Flickr API to output files

[1] https://github.com/naturalis/nbclassify/blob/master/nbclassify/scripts/nbc-harvest-images

A schematic overview is shown in figure 3.1. In order to use the Flickr API the folowing command line was used:

```
 python nbc-harvest-images.py  113733456@N06 meta.db -v harvest --page 6 pages 19 --per_page
500 images
```

113733456@N06 is the FlickrID that can be found in de url of the page of the map. After running the Flickr API there are to main outputs: the photo's and the .db-file. Figure 1.2 box 3 is were the Flickr API is used.

3.3 MonetDB

Figure 1.2 box 6 is were MonetDB is used.
MonetDB is an open-source database management system (DBMS). MonetDB is developed at the Centrum Wiskunde & Informatica in the Netherlands. On average MonetDB is monthly downloaded more than 10.000 times. MonetDB is designed for large databases and provides possibilities and infrastructures for large datasets. Because of innovation at all layers of a DBMS, MonetDB achieves greater speed compared to more traditional databases. MonetDB is a column-oriented database in contrast to the most databases. The most databases are row-oriented. Both row-oriented and column-oriented databases can use SQL(Structured Query Language: a programming language designed for managing data held in a database). An advantage of a column-oriented database is speed. Column-oriented databases boost performance by reducing the amount of data that needs to be read from the disk. In a row-oriented database all the rows have to be retrieved when a query is applied. In a column-oriented database only the columns in the query need to be retrieved, hence it is easy to upscale a column-oriented database [13] The architecture of Monetdb is divided in three layers. Each of these layers has got its own set of optimizers. The first layer provides a query interface for SQL . The SQL is transformed to MonetDB Assembly Language (MAL) which are passed to the next layer. In the second layer there are some cost based optimizers for the MAL. The third layer provides access to the data in the database. MonetDB is used a lot in telecommunication due to the ability that the number of columns per table is practically unlimited. MonetDB supports the SQL:2003 standard as well as application programming interfaces for programming languages such as C, Python, PHP and R. Therefor it is possible to run directly inside the database. [12]

3.4 Virtual server

Our supervisor gave us a virtual machine to work on. The database and images from flickr are saved on this virtual machine. The virtual machine is equipped with an i7 quadcore processor, 16 Gb RAM and 500 Gb disk space.

3.5 HOG features

Figure 1.2 box 8 is were HOG is used.
After the preprocessing, feature extraction of the ROI will be done using Histogram of Oriented Gradients (HOG). HOG is a dense feature method for images. HOG is trying to identfy shapes of structures in the image. The ROI is divided into an amount of equally sized cells. Each cell contains 8 by 8 pixels. 8 by 8 pixels was chosen because the cropped ROI is 256px by 256px and thus easily devided in these cells . HOG is a compact way to represent the image. The 8 by 8 cell is saved per pixel according the RGB (red green blue) principle. RGB color model is a way of saving the colours of an image through combining the colors red green and blue. Per color a pixel has a value depending on the pixel depth. An eight bit depth has got 2^n 256 colours. Nowadays eight bit depth is the norm a camera. For one pixel RGB needs 3 bits of information. HOG uses the gradient to save a bit. A gradient is a directional change in the intensity or color in an image. HOG only needs 2 bits per pixel.

The gradient of such a pixel contains 2 values per pixel, magnitude and direction.

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = arctan\frac{g_y}{g_x}$$

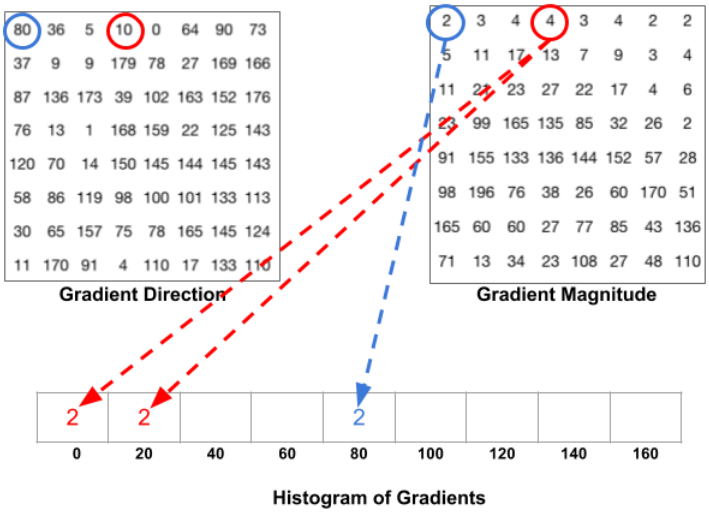

Figure 3.2: HOG voting per cell [15]

For each pixel in an 8 by 8 cell the magnitude and direction will be calculated. Using these numbers. A histogram of gradients is build for the 8 by 8 cell. In the histogram a set number of directions is used, 0, 20, 40, 60, 80, 100, 120, 140, 160. For each cell we are going to vote a degree. Figure 3.2 shows an example how the voting is done. For example, the upper left pixel (blue circle) is 80 degrees and has a magnitude of 2. Because 80 degrees is a set degree in the histogram we can vote 2 times for 80 degrees. The pixel marked with the red circle has a degree of 10. 10 is not one of the set degree so we have to divide the magnitude to 2 degrees. 10 is exactly in between 0 en 20 so we can split the 4 into 2 times 2 and vote 2 times for 0 degree and 2 times for 20 degree.

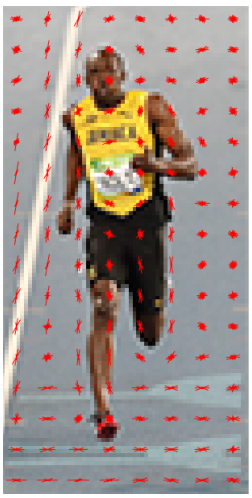

Figure 3.3: HOG visualized on an image [15]

This has to be done 64 times to fill the histogram. After accumulating all the votes we have 9 lines with a corresponding length. These lines are drawn on the 8 by 8 cell. This is the HOG feature of 1 cell. On figure 3.3 a visualization of the hog is presented. At each 8 by 8 cell are a number of vectors with corresponding degree (direction) of the histogram and length(magnitude). [14]

3.6 Machine learning

Machine Learning gives the computers the ability to learn without being explicitly programmed. The study of machine learning is evolved from computational learning theory and pattern recognition in artificial intelligence. The first sing of machine learning dates back to 1950. Alan Turing created the Turing Test. The goal of this test is to find out if a computer has real intelligence. A computer can pass this test if it is able to pretend to be a real human being. A real human has to believe the machine is also a human. In 1952 Arthur Samuel wrote the first program that can be considered as machine learning. The program was able to analyze and improve at checkers. The more the program played checkers, the better it would get at making up winning strategies and include those moves into the program. In 1957 the first neural network for computers was designed by Frank Rosenblatt. It was a simulation of the thought process of the human brain. In 1967 The first nearest neighbour algorithm was build. This was the first time a computer could use very basic pattern recognition. 2011 was an important year for machine learning. In this year IBM tried to beat the best Jeopardy (a quiz game) players in the world with a computer. The computer won easily. Machine learning is a hot topic in the past couple of years. This is due to the fact that big data is one of the hottest trends at the moment. Machine learning can be extremely good at making predictions or calculate suggestions on big data. A good example of a machine learning algorithm is implemented in Netflix. In Netflix there is a suggested for you area, in this section a couple of movies are suggested based on the movies the watcher has watched in the past. To make the computer able to solve problems it is necessary to train the computer. This can be done in three ways: (1) supervised learning, (2) unsupervised learning and (3) reinforcement learning.

1. Supervised learning: The computers gets example inputs and their outputs. When enough examples are presented the computer should learn a ”general rule” to match the in- and outputs.

2. Unsupervised learning: all the input data do not contain any labels. The main goals are to find patterns in the data or to find features. This can be helpful to find ”hidden” information.

3. Reinforcement learning: The computer has to learn a specified task. To learn the task, the computer gets feedback. With penaltys the computer is forced to perform the task in a good way [19].

3.6.1 IBk

The IBk algorithm is an implementation of the k-nearest neighbor algorithm (KNN). The k-nearest neighbor algorithm can be used for classification and regression. k is any number above 0. The output of the algorithm is a class label. During the training phase the feature vectors and class labels are being saved in a 2d space. During the classification phase a new feature vector without label will be plotted in the 2d space. At this point the algorithm calculates what the k nearest neighbors are. Image 3.4 shows an example of the KNN algorithm. The left graph shows five blue dots and six yellow dots. The with dot is an unlabeled dot and the algorithm has to label the with dot with blue or yellow. In this example k=3 is used. The algorithm calculates what the three nearest neighbors are. In this case 2 of them are yellow and one is blue thus the white point will be classified as yellow. [20]

In order to measure the distance between two points the Euclides formula is used which is shown below [23].

$$d(p,q) = d(q,p)\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

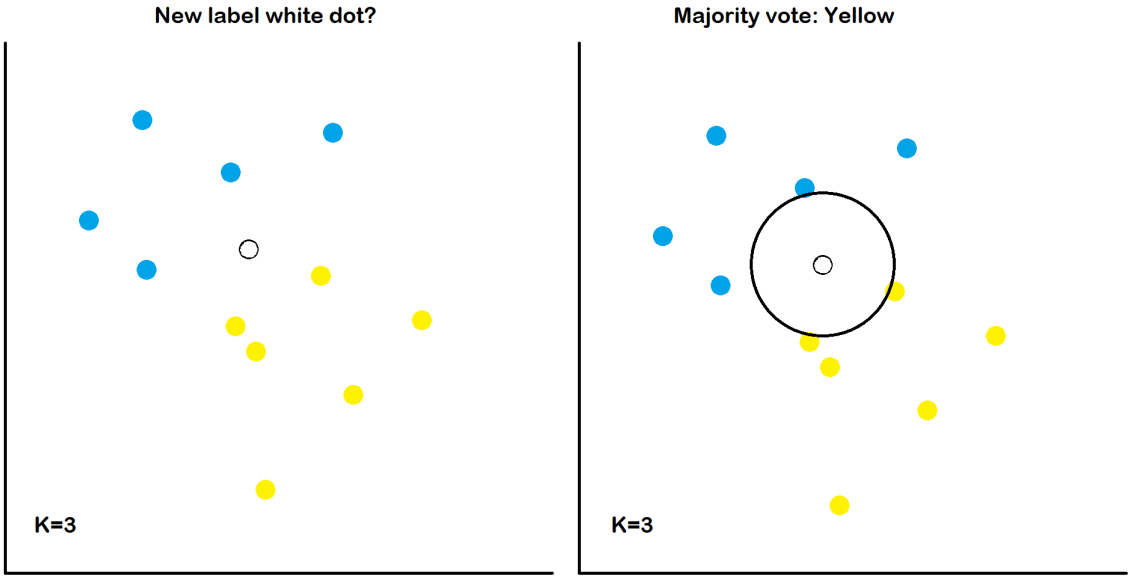

Figure 3.4: k-Nearest Neighbour example

3.6.2 Naïve Bayes

Naïve Bayes classifiers are classifiers that are based on the Bayes theorem. The Bayes theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event.
The simple statement of Bayes' theorem is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes classifiers are a family of simple probabilistic classifiers. These classifiers are based upon the Bayes theorem. Naive Bayes can both be used for binary classification problems and multi-class classification problems. Naïve Bayes works on conditional probability (the probability something will happen given the fact that something else has already occurred). Naïve Bayes bayes predicts membership probabilities for each class. Using these probabilities it is possible to calculate the probability of an event (class). [24]

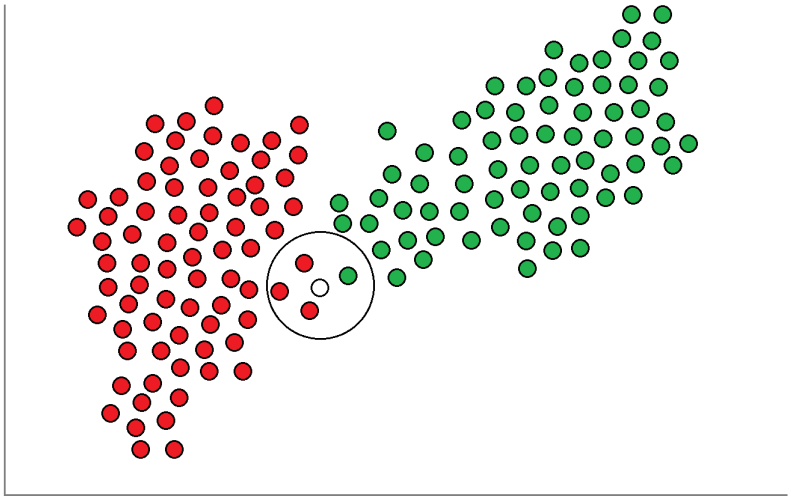

Figure 3.5: Naïve Bayes

Figure 3.5 shows an example to explain naïve Bayes. The data set contains labeled data (red and green dots) and unlabeled data (white dot). In the data set there are twice as many green dots (40) as red dots (20). If a new unlabeled data point is added the probability of the new object being green is 40/60 and red 20/60. The dots are well clustered so it is reasonable to assume the more Red/Green dots are near the new dot the higher the chance the new dot belongs to that particular dot.

When the new white dot is added a circle is drawn around the point. In this circle there are three red dots and one green dot. The probability of White being green is 1/40 and red is 3/20.

Using the Bayes' rule these two probabilities will be multiplied.

Likelihood of white dot being green: 40/60 * 1/40 = 1/60
Likelihood of white dot being red : 20/60 * 3/20 = 1/20

According to the naïve Bayes the white dot will be red.

3.6.3 Support Vector machine

Support vector machine (SVM) is an algorithm that builds a model on the basis of labeled data, with this model the SVM can classify unlabeled data.

There are 2 ways of computing such a model:

- Linear SVM
- Non-Linear SVM

Linear SVM is often used for binary classification problems. Image 3.6 shows a set of white and black dots. These are the labeled data points. The SVM searches for a hyperplane that divides thse two data sets. A lot of hyperplanes divide these two data sets. SVM searches for the hyperplane with the largest margin between the two classes. The hyperplane with maximal distance to the two nearest datapoints on each side is chosen. In this case h3 would be the best hyperplane. There are other ways of calculating the best hyperplan such as hard margin and soft margin.

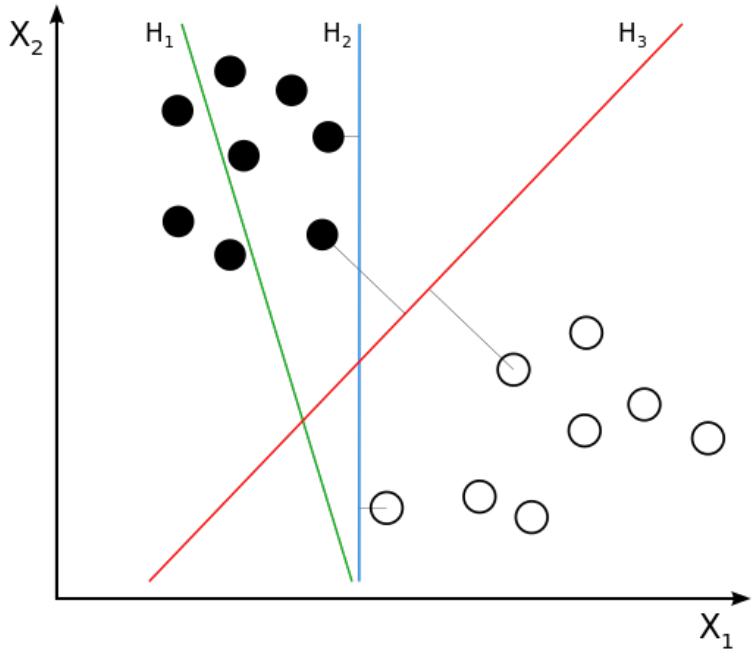

Figure 3.6: Linear SVM [21]

If the data does not allow classification by a linear SVM a non linear SVM can be used. If the data is not linear separable the data can be maped in a higher dimension and then a linear classifier can be used in this higher dimension. In the first data set the hyperplane will be nonlinear. The top data set in figure 3.7 is just simply classified by a linear SVM. The middle data set can't be classified by a linear SVM but when mapped in a higher dimensional space (bottom data set) it can be separated. The kernel trick is a efficient way of doing the mapping to a higher dimensional space. [25]

SVM were first designed for two-class classifiers. There are however ways to do multiclass classification with SVM. The one-versus-all (OVA) method can be used. OVA turns the multiclass problem in K binary problems

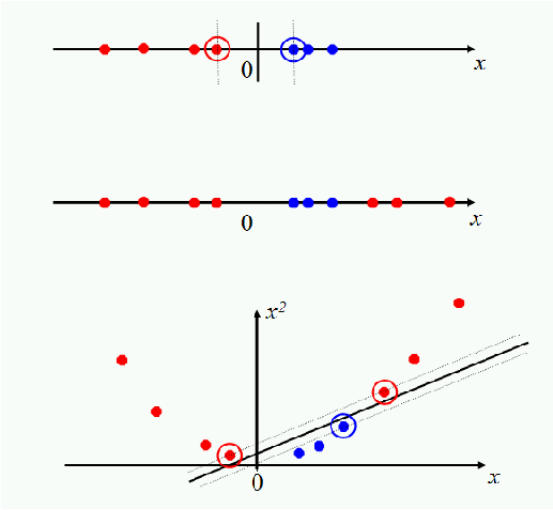

Figure 3.7: Nonlinear SVM [22]

(where k is the number of classes). If, for example, there are three classes, A, B and C. The OVA turn this in three binary problems. First A vs $B \cup C$, secondly B vs $A \cup C$ and lastly C vs $A \cup B$.

When a new unlabeled data point is added the point will go through the 3 classifiers. The first classifier calculates the odds of the new point being an A. The second classifier calculates the odds of the new point being a B (same goes for C). The class with the highest odds will be chosen in the prediction. [26]

3.7 Weka

Waikato environment for Knowledge Analysis (WEKA) is a datamining software written in java. Weka has some datamining algorithms build in. It is easy in use and will be a good starting point in finding the best algorithm. [17] Weka was used for the pilot testing as it provides rapid prototyping.

3.7.1 Comma-Separated Values

In computing a Comma-Separated Values (CSV) is a file where numbers and text are stored in plain text in tabular way (file extension .csv). Each row is a data record expect for the first row. The first row contains the column names. Known as header. Each data record is separated by a comma hence the name. The end of a row does not have a comma but ends at the end of the line (ascii 13) .
CSV was already used in the early days of business computing and is still used to pass data between computers. CSV is widely used but has a downside. It can not handle data if the data itself has got commas in it [16]. In our data dots are used so we do not have to take this problem into account.

3.7.2 Attribute-Relation File Format

An Attribute-Relation File Format (ARFF) file is a text file that is used in the datamining program WEKA. It looks a lot like a CSV-file but with some differences. As in CSV the ARFF file the data is separated by a comma. The difference between these two files is the way you need to write the attributes down. In CSV the first row contains the attributes. In ARFF the file start with the name of the relation, after that all the attributes are stated with the name of the attribute and the kind (NUMERIC OR CLASS). [17]

```
@RELATION Lepidoptera
@ATTRIBUTE hogfeature1  NUMERIC
@ATTRIBUTE hogfeature2  NUMERIC
@ATTRIBUTE hogfeature3  NUMERIC
@ATTRIBUTE hogfeature4  NUMERIC
@ATTRIBUTE class        {vulgaris, coon, nox, adonarensis, demolion}
```

This is an example of the header of an ARFF file. The HOG features are represented as attributes. Each attribute contains a number from the HOG-vector. The last attribute is the class attribute. In this attribute all the different data points have to be stated.

```
@DATA
5.1,3.5,1.4,0.2,vulgaris
4.9,3.0,1.4,0.2,coon
4.7,3.2,1.3,0.2,adonarensis
4.6,3.1,1.5,0.2,demolion
5.0,3.6,1.4,0.2,demolion
5.4,3.9,1.7,0.4,vulgaris
4.6,3.4,1.4,0.3,andonarensis
5.0,3.4,1.5,0.2,nox
4.4,2.9,1.4,0.2,coon
```

The data part of the ARFF file contains the data separated by commas. In this simplified example it contains four data points as there are four numeric attributes. The latest value in a row is the class attribute.

3.7.3 F_1 score

	Predicted		
Actual		True	False
	True	TN = 50	FP = 10
	False	FN = 5	TP = 100

Table 3.1: Confusion Matrix

There are four different terms that are important in a confusion matrix:

- False negatives (FN). The classifier predicted no but in reality they do have the disease.
- False positives (FP). The classifier predicted yes but in reality they do not have the disease.
- True negatives (TN). The classifier predicted no and predicted this correctly
- True positives (TP). The classifier predicted yes and predicted this correctly.

Table 3.1 shows a Confusion matrix. This matrix described the frequency of the four combinations of subgroup and target. In this example there are 165 predictions. The predictions are whether or not someone is diseased. Out of the 165 predictions the classifier predicted no 55 times and yes 110 times. In reality 60 people did not have the disease and 105 did had the disease.

FN and FP are the two kind of errors. It depends on the situation which error is worse. For example in case of a doctor with a patient who is tested for cancer a FN is worse than a FP. FP will lead to unnecessary treatment but FN will lead to false diagnostic. In case of a severe disease this could lead to death. In case of a criminal court it could be considered preferable to make a FN error. In case of a FP error a innocent person is locked up for years of their life. A FN error would mean a guilty person walking free.

The F_1 score is a method to measure the accuracy of a test. The F_1 score uses the precision (p) and the recall (r) to compute a score.

The precision is the correct positive results divided by the total number of positive results:

$$Precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{\{retrieved\ documents\}}$$

The recall is the correct positive results divided by the total number of positive results that should have been returned.

$$Recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{\{relevant\ documents\}}$$

The F_1 score is the harmonic average of the precision and recall. $1 \geq F_1 \geq 0$ where one is the best value and zero the worst.

The formula used to calculate the F_1 score:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

[18]

Chapter 4

Implementation

In this chapter the implementation of the Materials and Methods (chapter 3) will be discussed.

4.1 Harvest

The first step in creating a pipeline for the images is importing the images from Flickr. As earlier stated a Flickr harvester was retrieved from the the Naturalis' GitHub-page [1]. From the Flickr page of Naturalis 1842 butterflies where imported to the project server. The images were divided in several maps. First the images were divided on the basis of genus, secondly on the basis of species.

The images are saved with the file name of the Flickr image. A number which is not related to the image of the butterfly. Once all the images were imported to the server the next step would be is connecting additional information about the butterflies with the pictures. This will be done by filling the database, MonetDB. The images are stored in folders on the server and the corresponding url of each image is stored together with the metadata in MonetDB.

Except harvesting the images the program had another function. During harvesting the images the program created a .db file with all the tags of the pictures in it. This was useless at first because in the .db-file there was not stated which metadate belongs to which image. So it was not possible to link the .db file to the images. After a few weeks Naturalis sends a file which linked the file name to the registration number assigned by Naturalis (ZMA.INS.XXXXXXX where X is a number). With this file it was possible to link the .db file to the images from Flickr.

With the .db file and the external file from Naturalis the extra information of the images was linked. A small program was written that would put these information into the database. To connect the image with this information a link to the image is also stored in MonetDB.

[1] https://github.com/naturalis/nbclassify/blob/master/nbclassify/scripts/nbc-harvest-images

With this new file the MonetDB was filled with all the tags from Flickr. At this point the database contained the following attributes as shown in Table 4.1.

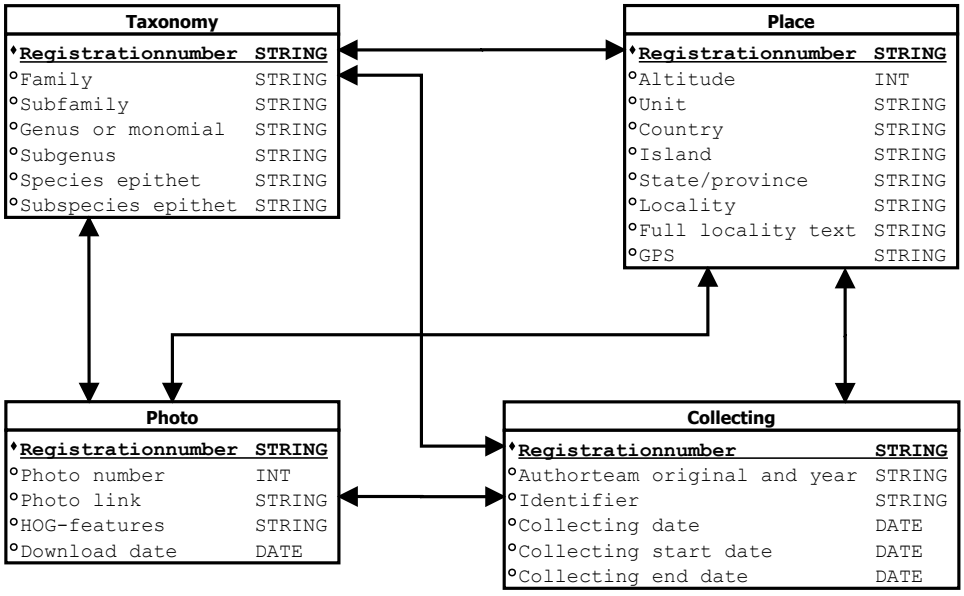

Figure 4.1: The database tables

- Registrationnumber	- Parenthesis	- Island
- Family	- Identifier	- State/province
- Subfamily	- Altitude	- Locality
- Genus or monomial	- Unit	- Full locality text
- Subgenus	- Collecting date	- Link to photo
- Species epithet	- Collecting start date	- Photo number
- Subspecies epithet	- Collecting end date	- HOG-feature
- Authorteam original and year	- Country	

Table 4.1: Attributes in the database

When all the attributes were clear it was possible to split them in several tables. In this case the data is split into four tables: Taxonomy, Place, Collecting, and Photo. This is visualized in figure 4.1. The first table, Taxonomy, is a taxonomic representation of the butterflies. The next table, Place contains information about where the butterfly is found. Linking this table to the Collecting table it is possible to figure out when and who collected the butterfly. At last there is a place needed to store the photo and its features. This is done in the photo table. In the design phase it was clear all the information stored in the database is linked in one item: the butterfly itself. Therefore the decision is made to link every table with the Registrationnumber as primary key.

The split in four table is done for clarity and speed. As the most databases also MonetDB uses Structured Query Language (SQL) for executing actions. Every table presents one specified theme about a butterfly in the database. Splitting a table with a lot of columns into more table can speed up the query process. Instead of looping trough all the columns to search what you are looking for, it is faster to combine several tables with an subclause. For example:

```
SELECT T.Species epithet, P.Island FROM Taxonomy T, Place P
WHERE T.Registrationnumber = P.Registrationnumber
```

The result will be the specie and island where it lives. This means the query has to look at two tables instead of one big table. The search querry can skip all other information in the other tables which speeds up the result, because selecting a few columns in a table costs more search time, than smaller tables combined with clauses. This also helps for the clarity for the user.

4.2 Preprocessing

M. Li build a preprocess program in her earlier research. M. Li had written this program in Matlab. Because of the lack of knowledge in matlab and the fact that all the comments in the program were written Chinese the program was not very handy in the first place. The second reason to start over again is to make it possible to use the program directly inside MonetDB, which supports Python. Querying, feature extracting, preprocessing and learning algorithms can be linked together in one single language. Therefor Python would be a better solution instead of matlabcode. However the pipeline she used in her program was usable. Because the matlabcode of Mengke was not very handy a new program was written on the basis of the pipeline of Mengke her program. The goal of the program is to get firstly the ROI and secondly the HOG features. For the usability inside MonetDB and cohesion with other programs that will be used in this research, all the programs are written in python.

The first thing the program has to do is resize every image. The images of the butterflies are taken with the same camera but are saved in different sizes. Secondly the program cuts out a square bounding box around the butterfly. This box is set that the whole butterfly fit in the box. At this point a box with butterfly is left. The box measures 128*128 pixels. The box has got some usefully information (butterfly) as well as redundant information (background). Because only the butterfly is needed, the background should be removed. A way of separating the butterfly from the background is using a mask.

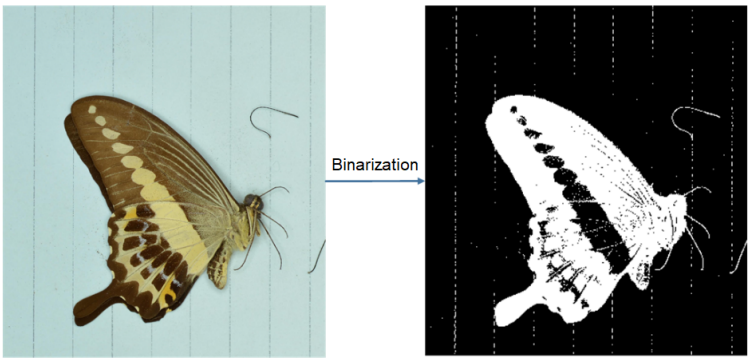

Figure 4.2: binarization of the image [9]

The first step of getting the mask is Image binarization. A binary image is an image that only has two values for each pixel, 0 and 1 represented by black and white respectively, but any two colours can be used. In figure 4.2 there is an example of image binarization.

The binary image that is produced is almost a mask. The only problem is the black holes within the butterfly. In order to fill the holes inside the butterfly the program uses a closing operation. This operation will fill in all the holes within the main object in the picture. Also noise from the background will be eliminated.

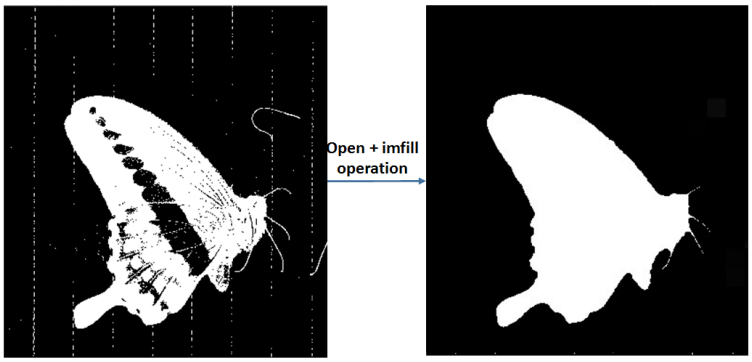

Figure 4.3: Filling the image and reduce the noise [9]

Figure 4.3 shows the rectangle with the butterfly before and after the filling operation.

Figure 4.4 shows the butterfly with the mask. The mask is a binary image where each pixel has a value of one or zero. This mask is laid over the initial rectangle. The pixels that are value zero will stay value zero. The pixels that are value one will be the pixel of the initial rectangle. At the end of the first part of the program the full Flickr image will be preprocessed to the ROI.

Figure 4.4: applying the mask [9]

4.2.1 HOG features

With the ROI it is now possible to start feature extraction. Calculating the hog features of an image will be done in python program. The output this program will be a vector with 2048 numbers. Namely, each images is split by the HOG program in 16x16 blocks, where each block contains 8 cells. This means $16 * 16 * 8 = 2048$ numbers in the vector. Futhermore the HOG feature uses grayscale. This means every cells is represented in one number $16 * 16 * 8 * 1 = 2048$ (the grayscale). For extension this can be extended to RGB. This means 3 numbers instead of 1 which results in $16 * 16 * 8 * 3 = 6144$ numbers. This leads to three times bigger vector to represent the image. For now this is not necessary as 2048 numbers can be enough for the learning algorithms to split the dataset in several species.

This 2048 numbers big vector represents the butterfly. Figure 4.5 shows a visual representation of the HOG features. Where the image is just a black picture there is no gradient. Therefor the HOG values are zero here.

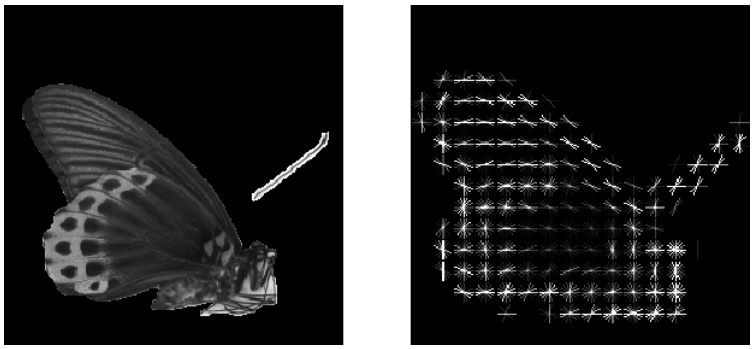

Figure 4.5: Left: the image in gray-scale, right: the HOG feature visualized

4.2.2 Converting for WEKA

The second part of the program is possible to convert the images to HOG features. It exports a .csv file with the 2048 numbers for each image linked to the image name. Below an example of a the csv-file. The hog1 till hog2048 are here simplified to hog1 till hog4 for readability.

```
hog1, hog2, hog3, hog4, foto
0.01, 0.02, 0.00, 0.02, vulgaris
0.00, 0.01, 0.01, 0.01, coon
```

This will be the labeled dataset.

As explained in chapter 3 WEKA will be used for experiments and testing. The file is exported as a .csv file but WEKA only accepts .arff files. On the internet a csv-to-arff converter [2] can be found. Using this converter the data will be ready for WEKA classifiers.

4.2.3 Weka

With the .arff files it is possible to use Weka. Weka will be used to find the best classifier. The algorithms that will be tested are: iBk, naïve bayes and Support Vector Machine. The best algorithm will be chosen on base of the F_1 score. The F_1 score will be calculating with the use of k-fold cross-validation. K-fold cross validation randomly divides the sample set in k different sub samples. In the test cases there is made use of 10 folds cross-validation. One of these sub samples is used as the test set, the other k-1 sub samples are used as training set. This process is repeated k times. Every subset is one time the training set. [27]

[2] https://github.com/christinequintana/CSV-to-ARFF

Chapter 5

Results

5.1 Dataset

After using the Flickr image harvester the database contained 1842 images of butterflies. Now it is time to clean the dataset. Befor the data can be used in a proper way, discarding the dataset from noise is needed. There are two kinds of noise. The first kind of noise are images without a label, which make them unusable training. The second kind of noise are images that have photographed butterflies in the wrong manner. For the most part the butterflies are photographed as seen in figure 2.1. The butterfly is fold in half. Some of the butterflies are not folded and photographed with two wings instead of one (see figure 5.1) . When extracting the features this yield other and more values. In that case the train data would not be the same for every butterfly. After removing this erroneous sets the database contained 1753 butterflies.

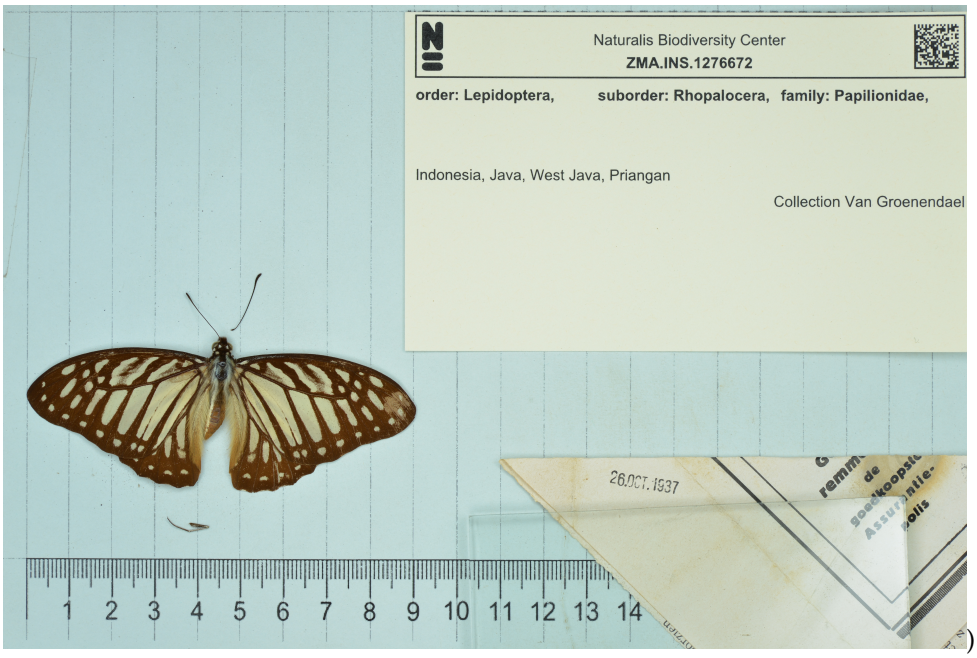

Figure 5.1: Noise

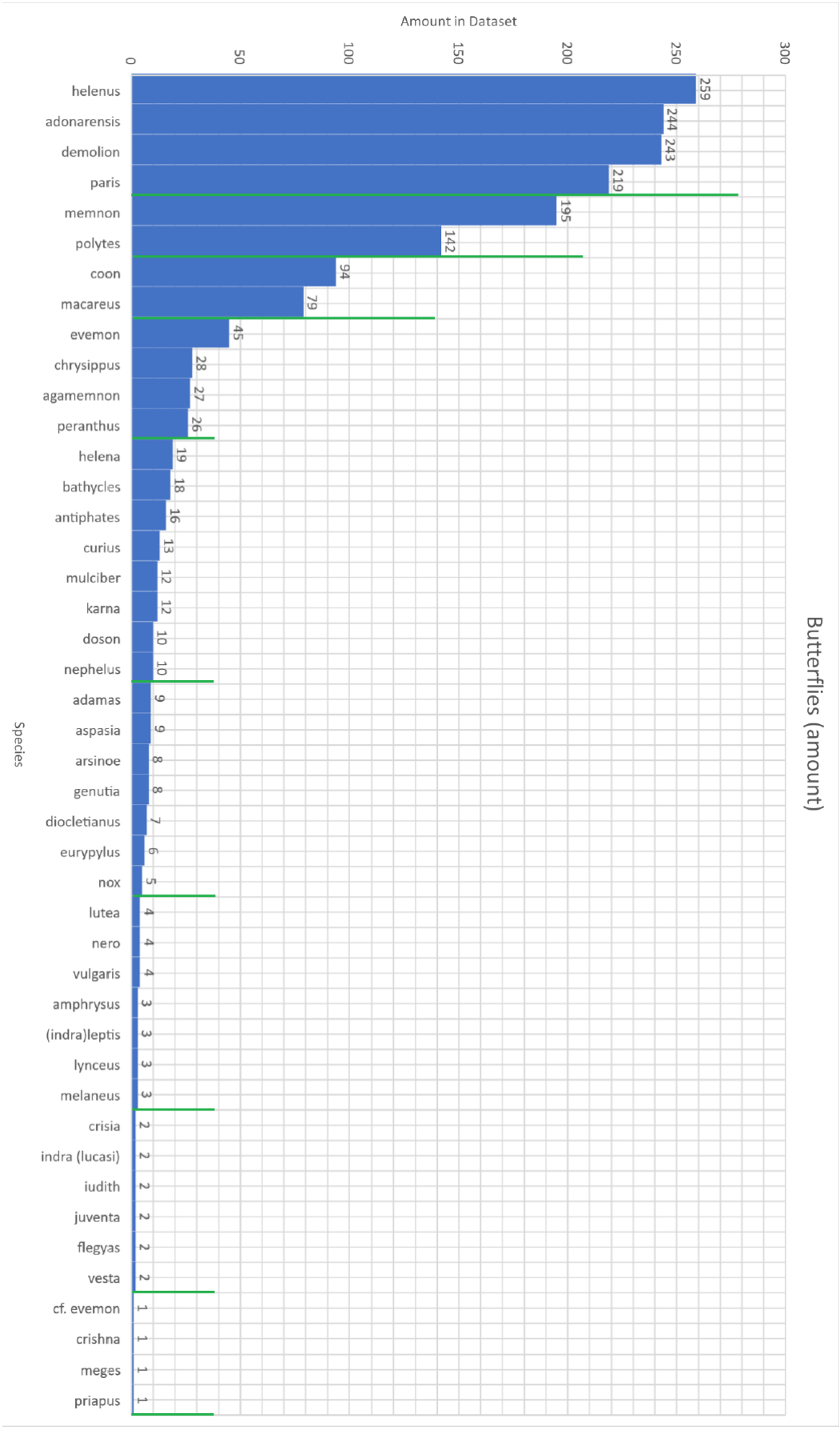

Figure 5.2: Histogram of the examples per specie

5.2 Boundary conditions

Figure 5.2 shows the amount of examples per species. In order for a classifier to work it needs at least 2 examples per class and preferably more. One example for the training set (to train the algorithm) and one in the test set (to test the algorithm). In the dataset from Naturalis there are 4 species with only one example: cf. evemon, chrishna, meges and priapus. For these species adding data is important so later it would be possible to use them for training. Now with only one example, these species can not be tested. Therefor these species will not be used in the experiments.

	N>0	N>1	N>2	N>4	N>9	N>24	N>49	N>74	N>99	N>199
Naïve bayes	0.219	0.218	0.229	0.254	0.273	0.300	0.365	0.362	0.567	0.634
KNN	0.606	0.618	0.619	0.680	0.702	0.731	0.754	0.741	0.754	0.802
SVM	0.669	0.668	0.676	0.719	0.726	0.750	0.763	0.750	0.753	0.796

Table 5.1: F_1 scores

Table 5.1 shows the F_1 score of the different classifiers. The N represents the number of examples per species. The vertical lines in figure 5.2 indicate where the dataset is split. So, when N>3 all the species with less than 4 examples are not taken into account. The F_1 score is calculated on the basis of 10 fold cross validation. It is possible to change some parameters in WEKA. Starting these tests all parameters where set to default. Some first tests to check everything works correctly and finding out which parameter leads to different scores. The decision is made to use the linear SVM, because this is the most simple function and offers great speed to performance. Of course it is possible to change it to non-linear functions. This can help to improve the results but costs lot of extra time to execute. For this test all these parameters where kept the same except for the KNN classifier. Adjusting the k for the same train and test set shows a k of 4 performs best. So for the KNN classifier a k of four was used. With naïve base every parameter was set as default, because changing the parameters did not show improvement in the results.

5.3 Algorithms

The first experiment is done on the complete dataset. N represents the number of examples per species that have to be at least in the training set. So for example N >3 all species with less examples than 4 are not taken into account. But this means there is no equal distribution because some species have more examples than others. Also there are some species which have only a few examples to train on. With this in mind it can happen that all the species with less training data cannot be classified in a simple way and they can influence the final F_1 scores a lot. Therefor there is need for another experiment that has a proportionate distribution over all the species. This will be experiment two. The results of experiment one can be found in table 5.1.

Table 5.2 shows the result of the second experiment. The goal of the second experiment was to find the best classifying algorithm. Now there is made use of proportionate distribution. In contrast to experiment one all the species has at least 10 instances to run the algorithm on. Later this will be raised to 140 instances for each

N	Naïve Bayes	Ibk	SVM
10	0.625	0.667	0.861
20	0.570	0.780	0.894
30	0.644	0.843	0.937
40	0.704	0.839	0.915
50	0.717	0.856	0.912
60	0.696	0.849	0.941
70	0.681	0.867	0.959
80	0.686	0.856	0.945
90	0.692	0.866	0.949
100	0.693	0.862	0.932
110	0.694	0.857	0.932
120	0.692	0.858	0.925
130	0.679	0.845	0.901
140	0.673	0.821	0.875

Table 5.2: F1 scores

specie. For this experiment only the species with more than 100 examples per species were used. In order to test which classification algorithm is the best, the only thing that changed during the experiment was the amount of instances per specie. The N represents the exact number of instances per specie taken for the test. For example this means when $N = 10$, there are 10 instances of each specie in the set. With 6 species there are 60 $(10 * 6)$ instances in total. For $N = 20$ there are 120 $(20 * 6)$ instances in total and so on. For the three algorithms the F_1 score was computed which are shown in figure 5.2.

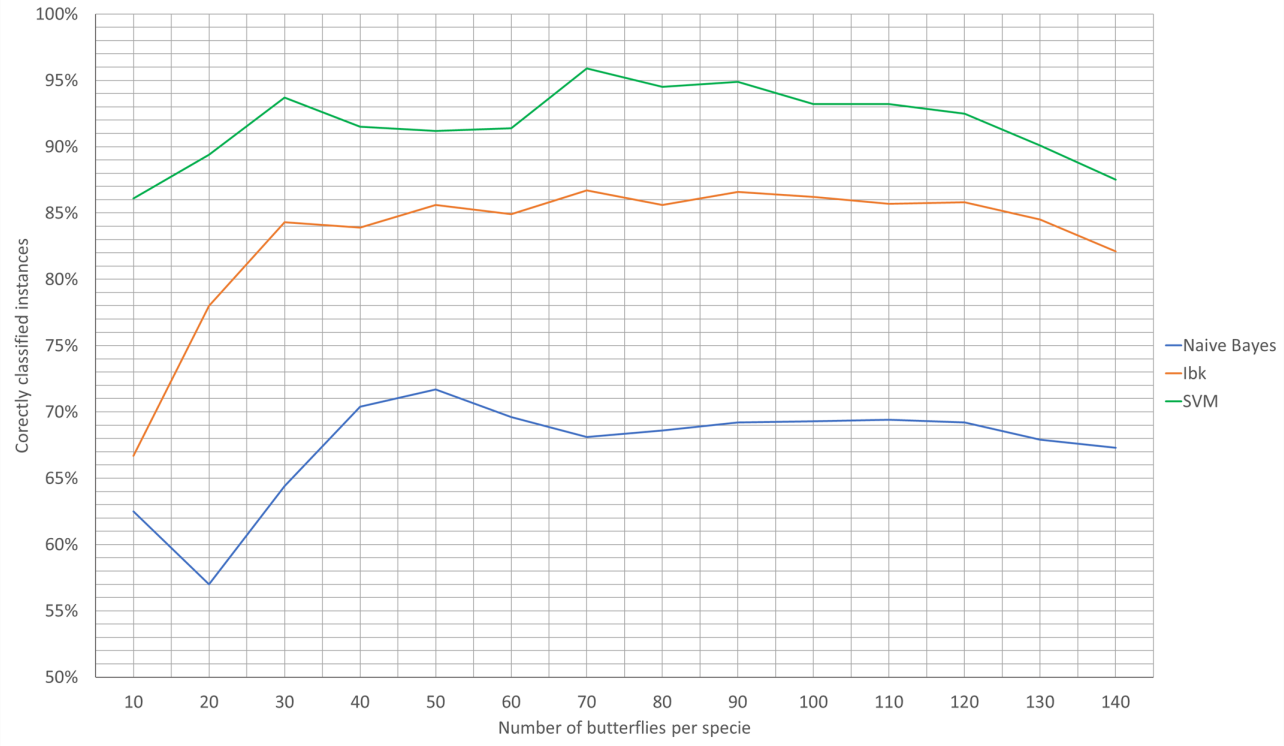

Figure 5.3: performance

In the table are the F_1 scores of correctly classified instances using a 10-fold cross validation. In order to find this and being able to scale the amount of butterflies per specie, a data set that contained only butterflies with

more than 140 examples per species is used. The original data set includes six butterflies with N >140. Per specie an equal amount of examples where put into the data set used for this experiment.

The results are plotted in figure 5.3. This figure shows great difference between the performance of the three classification algorithms. The upper line is the performance of the SVM classifier. It outperforms the other two classifiers with about 5% difference. The KNN classifier (ibk) performs a bit less, but still around 85%. The worst classifier in this comparison is the Naïve Bayes algorithm, which performance is between 68% and 72%.

Chapter 6

Conclusions /Discussion /Future work

In this chapter the results of chapter 5 will be discussed and conclusions will be drawn. Further more the thesis will be discussed and there will be a glance at the future.

6.1 Conclusions

6.1.1 Is MonetDB qualified for this problem?

There are two characteristics that needs to be taken into account with the problem: scalability and speed. More and more butterflies are going to be digitized in the future, hence scalabilitiy is important. MonetDB is suited for a larger scale. MonetDB has a scalable architecture, via vertical data organization en in-memory optimization MonetDB guarantees fast query responses for small and very large datasets. These fast query responses are established without substantial hardware investments.

Also MonetDB offers great support to program languages such as Python and R. With this possibility it is possible to run programs directly inside the database. Instead of a lot client-server traffic when running a program outside the database, the only traffic will be the output of the program. This speeds up the process. Besides, if the most used feature will be adding butterflies in the same way and training the classifiers model, then this can also be done with a stored procedure. When using a lot of code or replaying some mutations regularly, stored procedures can be faster than repeatedly SQL-queries.

6.1.2 Can the classification process be automated?

During this thesis the classification process (including the harvesting, feature extraction and more) was not fully automated. For example there were different python programs for harvesting and feature extraction. The only programming code used during this thesis is Python. It is possible to combine all these different Python

code so it will be executed as one automated script. So eventually this script will do the full classification process automated. The user has to upload the path to the new images and the program will output everything in the MonetDB.

6.1.3 Is it possible to classify the Lepidoptera on species level?

With the data set currently available it is not possible to classify all butterflies on specie level. The main problem with the current data set is the amount of butterflies per specie in it. Some species only occur once. It is not possible for any classifier to classify a specie with only on example. Table 5.2 shows the results of the species with more than 100 examples. With this data set the best classifier can give the correct specie name with 95% certainty

6.1.4 Are there any boundary conditions to this classifier?

There are some boundary conditions in order to classify the butterflies correctly. The amount of examples per species has to be greater than 40. In chapter 5 in figure 5.3 the best performance is between 40 and 90 instances per specie.

6.1.5 Which method is the most successful classifier?

The classifier that has got the highest percentage of correctly classified instances is SVM. Table 5.2 and figure 5.3 show that this classifier outperform the other two at every level of instances. But when the data set increases with more than 120 butterflies per specie the chart is sloping down. Within a specie there is a difference between the looks of male and female. When there are a lot of butterflies (male and female) in the dataset there will be deviation within one specie. Because of this deviation it could happen that the results decline.

6.2 Discussion

Still the dataset does not contain abundance of data. In experiment 2 there is only made use of six different species. This because there was no more data available. This research can be extended to view if much larger amounts of butterflies per specie can increase the outcome even further. Furthermore also other feature can be used to see if the outcome of the algorithms can be more optimized with extra information about every butterlfy.

Some of the results in table 5.2 are remarkable. It would be logical that the F_1 score would be positive correlated with the N, this because in general more training should be result in better performance. In this case it is not. The highest F_1 score corresponds not with the highest N. An explanation of these results could be that there was a deviation within the different species. Within species the male and female butterfly differ from each

otter. When adding more examples this deviation can be great enough to influence the classifier causing less generalization and therefore worse performance with a high N in experiment 2.

In this research the HOG-features are calculated with use of grayscale to make the data that discribes a butterfly as small as possible. To save more information into the HOG-vector the grayscale can be replaced by RGB. This will create a three times bigger vector to represent the butterfly. This can lead to better performance, alltough the executing time will take much longer.

For extending the pipeline it can be useful to split the butterflies per specie in male and female parts. This leads maybe to better performance. Male butterflies and female of the same specie can have great difference in shape and wing drawings. Therefor if this is taken into account the performance can even better than the now achieved 90-95%.

6.3 Future work

Currently the process of classification is cumbersome. The pipeline is usable but it is not user friendly. In order for the program to be useful for Naturalis an interface has to be build. There are two future scenario's discussed, both the scenario's need a different interface.

1. Naturalis continuous to store the images on Flickr.
2. Naturalis are going to store the images on a server.

6.3.1 Scenario 1

Naturalis is currently storing the images on Flickr. In order to classify the butterflies the images need to be harvested from Flickr. In this scenario an interface would be preferred where the user needs to fill in the FlickrID and needs to select the images the user want to harvest. After this step the program can go autonomous through the pipeline. After selecting the FlickrID and selecting the images the program will fill the database with the specie name and other tags such as file name. The program will also give a notification if a image can not be classified.

6.3.2 Scenario 2

Scenario 2 assumes a change in the current digitization process. If Naturalis saves the images on a dedicated server instead of uploading them to Flickr a step of the pipeline can be skipped. When the images are uploaded to the server the interface can be a drag and drop system. The new images can be selected (or a program can be designed that indicates if new images have been added) and dropped in the new interface. The program now walks through the pipeline and fills the database. Figure 6.1 shows the two pipelines. The upper pipeline applies to a scenario were the butterfly is not classified. With out program the butterfly will be classified and

put in the database. The yellow pipeline applies to the scenario were a butterlfy is already classified. The data of the butterfly will be stored in the database with all the information. The butterfly is already classified so it can be used to re-train the model.

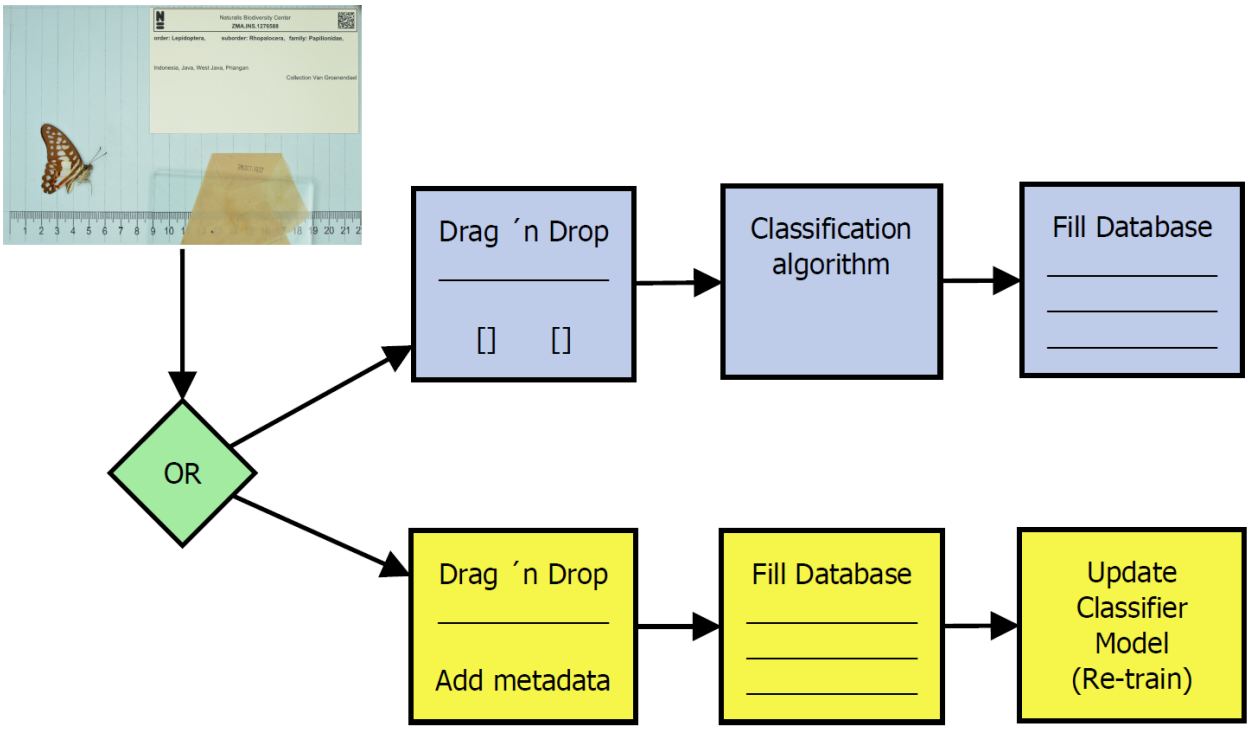

Figure 6.1: Two pipelines: classification or improve the classification model

Scenario 2 is preferred over scenario 1 because it saves time. Currently the images are stored on Flickr and need to be harvested to a server. It would save time to upload the images directly to the server.

6.3.3 Location

Another group of 2 students is currently working with the same data set. Their objective is to map the location where the butterflies are found. The database is divided into 4 database tables. One of the tables, which is called Place, contains all the information of this location. The other duo will map the butterflies. If this is done correctly it is possible to link the database and the map.

Other applications

The program is currently build around butterflies. As M. Li has shown a matlab program can also be build around other animals and Naturalis shows a program for flowers with its OrchID. All programs including the new python program for this research show that machine learning can do classification. If the database is large enough almost every animal can be classified as long as there are enough examples per species. How many examples depends on the learning algorithm and the amount of examples are also important to classify on deep levels. All the programs work with images in the same setup as data. Later on, another application could

be in video. Whenever someone makes a video of a flying butterfly and there is a frame with the wing on it the classifier would work as well.

Bibliography

[1] Lucking, Robert. *Taxonomy: a discipline on the brik of extiction. Are DNA barcode scanners the future of biodiversity*, 2008, Archives des Sciences 61, 75e88

[2] BISBY, Frank A., et al. *Species 2000 & ITIS Catalogue of Life*, Annual Checklist. 2006.

[3] `https://www.naturalis.nl/nl` last visited: 14/10/2017.

[4] `https://www.naturalis.nl/nl/over-ons/nieuws/blogs/Geologie/digitaliseren-naturalis-het-fes-project/` Visited: 15/10/2017

[5] `https://bruynzeel-storage.com/portfolio-item/naturalis-netherlands/` Visited: 15/10/2017

[6] Boero, Ferdinando. *Light after dark: the partnership for enhancing expertise in taxonomy*. Trends in Ecology & Evolution, 2001, 16.5: 266. Annalen der Physik, 322(10):891921, 1905.

[7] Cohen, H., & Lefebvre, C. (Eds.). *Handbook of Categorization in Cognitive Science*. Elsevier. 2005

[8] `http://orch-id.naturalis.nl/`Last visited: 10/01/2018

[9] Li, Mengke. *Taxonomy Algorithms of Javanese Butterflies Base on Machine Learning*

[10] `https://www.flickr.com/` last visited: 23/09/2017

[11] Van Rossum, Guido, et al.*Python Programming Language*. In: USENIX Annual Technical Conference. 2007. p. 36.

[12] Idreos, Stratos, et al. *MonetDB: Two decades of research in column-oriented database architectures*. A Quarterly Bulletin of the IEEE Computer Society Technical Committee on Database Engineering, 2012, 35.1: 40-45.

[13] Ventana; et al. *Ins and Outs of Columnar Databases*. 2011

[14] Onishi, Katsunori; Takiguchi, Tetsuya; Ariki, Yasuo. *3D human posture estimation using the HOG features from monocular image. In: Pattern Recognition*, 2008. ICPR 2008. 19th International Conference on. IEEE, 2008. p. 1-4.

[15] `https://www.learnopencv.com/histogram-of-oriented-gradients/` last visited: 7/02/2018

[16] Shafranovich, Yakov. *Common format and MIME type for comma-separated values (CSV) files*. 2005.

[17] Holmes, Geoffrey; Donkin, Andrew; Witten, Ian H. *Weka: A machine learning workbench*. In: Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on. IEEE, 1994. p. 357-361.

[18] Goutte, Cyril; Gaussier, Eric. *A probabilistic interpretation of precision, recall and F-score, with implication for evaluation*. In: ECIR. 2005. p. 345-359.

[19] Nasrabadi, Nasser M. *Pattern recognition and machine learning*. Journal of electronic imaging, 2007, 16.4: 049901.

[20] Larose, Daniel T. *Knearest neighbor algorithm. Discovering Knowledge in Data: An Introduction to Data Mining*, 2005, 90-106.

[21] `https://nl.wikipedia.org/wiki/Support_vector_machine` last visited 7/02/18

[22] `https://nlp.stanford.edu/IR-book/html/htmledition/nonlinear-svms-1.html` last visited 7/02/18

[23] Lehmer, Derrick H. *Euclid's algorithm for large numbers*. American Mathematical Monthly, 1938, 227-233.

[24] Mccallum, Andrew, et al. *A comparison of event models for naive bayes text classification*. In: AAAI-98 workshop on learning for text categorization. 1998. p. 41-48.

[25] Zheng, Jun, et al. *An online incremental learning support vector machine for large-scale data*. Neural Computing and Applications, 2013, 22.5: 1023-1035.

[26] Manning, Christopher. *Introduction to Information Retrieval*. 2008. p. 318-341.

[27] Bengio, Yoshua; Grandvalet, Yves. *No unbiased estimator of the variance of k-fold cross-validation*. Journal of machine learning research, 2004, 5.Sep: 1089-1105.